VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Dynamické generování mapové kompozice na webovém rozhraní

Dynamic Map Composition for Web-based User Interfaces

2017 Bc. Jakub Smékal

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Jakub Smékal**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Dynamické generování mapové kompozice na webovém rozhraní
Dynamic Map Composition for Web-based User Interfaces

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je dynamické generování interaktivní mapové kompozice na webovém rozhraní systému Floreon+ na základě aktuálně dostupných geografických služeb splňujících standardy OGC. Výsledné řešení bude generovat klientské skripty pro práci s mapovou kompozicí včetně automatického vytvoření interaktivního prvku pro ovládání zobrazených služeb. Součástí bude i návrh a implementace rozhraní interakce vygenerované mapové kompozice se zbytkem webového rozhraní.

Jednotlivé body zadání jsou:
1. Prostudování možností geografických služeb splňující standardy OGC.
2. Popis možností knihovny OpenLayers s cílením na využité prvky v rámci implementace.
3. Vytvoření funkčního kódu pro generování mapových komponent.
4. Úpravení webového rozhraní pro podporu navrženého způsobu interakce.
5. Diskuze nad možnostmi a omezeními vyvinutého řešení.
6. Testování a zhodnocení testů.

Seznam doporučené odborné literatury:

[1] OpenLayers, http://openlayers.org/en/v3.0.0/apidoc/
[2] GeoServer documentation, http://docs.geoserver.org
[3] Esri Support for Geospatial Standards: OGC and ISO/TC211, An Esri® White Paper, May 2015, http://www.esri.com/library/whitepapers/pdfs/supported-ogc-iso-standards.pdf
[4] Thomas Gratier, Paul Spencer, Erik Hazzard: OpenLayers 3: Beginner's Guide, 2015, PACKT

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017 .....

Rád bych poděkoval vývojářskému týmu systém Floreon+, jmenovitě pak zvláště Ing. Štěpánu Kuchařovi, Ing. Marku Bobrovi, Ing. Davidu Vojtkovi, PhD. a Bc. Janu Křenkovi. Největší dík však patří Ing. Janu Martinovičovi,Ph.D. neboť díky němu jsem dostal příležitost podílet se na tomto významném projektu.

Abstrakt

Systém Floreon+ je modulárním systémem pro podporu krizových řízení. Z původní myšlenky simulování povodňových situací se systém rozrostl do mnoha odvětví, jako například krizová řízení dopravy, simulace znečištění a také například analýza dopravních uzlů. S přibývající funkčností také narůstá počet mapových vrstev, které je nutné do systému integrovat a vhodně vizualizovat. Cílem této práce bylo vyvinout funkcionalitu, která umožní přidávat do systému mapové vrstvy dynamicky, a tudíž pro každého uživatele systému se sestaví mapová kompozice dle jeho oprávnění a nastavení. Výsledný systém je mnohem jednodušeji rozšiřitelný a s nižší potřebou zásahu do zdrojového kódu. Samotná práce je rozdělená do tří částí. První část popisuje architekturu systému a technologie, které jsou v implementaci využity. Druhá část popisuje stav systému před implementací dynamické kompozice a třetí část popisuje změny provedené v systému a samotný postup implementace.

Klíčová slova: Floreon, mapová kompozice, dynamické rozhraní, OpenLayers, interakce

Abstract

The Floreon+ system is a modular system for crisis management support. From the initial idea to simulate flood scenarios, the system has grown to include many areas, such as traffic crisis management, pollution simulation or traffic node analysis. With the increase in functionality also grows the number of map layers which must be integrated into the system and appropriately visualised. The aim of this thesis was to develop a feature which would allow map layers to be added to the system dynamically, and hence compile a map composition for each user depending on their permissions and settings. The final system is much simpler to extend and requires fewer changes of the source code. The thesis itself is split into three parts: the first part describes the architecture of the system and the technologies used in the implementation. The second part describes the state of the system before dynamic composition is implemented, and the third part describes changes made to system and the actual implementation method.

Key Words: Floreon, map composition, dynamic interface, OpenLayers, interaction

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous Javascript and XML
API	–	Application Programming Interface
CQL	–	Common Query Language
CSS	–	Cascading Style Sheet
CSV	–	Comma Separated Values
DOM	–	Document Object Model
DTO	–	Data transfer object
GML	–	Geography Markup Language
HTML	–	HyperText Markup Language
JSON	–	JavaScript Object Notation
OGC	–	Open Geospatial Consortium
RDBMS	–	Relational DataBase Management System
WCF	–	Windows Communication Foundation
WebGL	–	Web Graphics Library
WFS	–	Web Feature Service
WMS	–	Web Map Service
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Systém Floreon+ se vyvíjí ve vysokoškolském ústavu IT4Innovations, Národním superpočítačovém centru při VŠB-TUO, jako nástroj sloužící k pomoci s řízením krizových situací. Tomuto projektu se věnuje předně tým „ADAS“, označovaný také Floreon+, a to v rámci Laboratoře pro náročné datové analýzy a simulace.

Floreon+ vznikl hlavně za účelem vytvoření integrační a provozní platformy pro monitorování, modelování, predikci a podporu řešení krizových situací především v oblasti Moravskoslezského kraje. Ačkoliv nosným tématem projektu je hydrologie, dochází v dnešní době k integraci i dalších oblastech, jako jsou znečištění životního prostředí nebo problematika monitorování a predikcí dopravních situací (propojení se systémem RODOS[6]). A právě z důvodu neustálého nárůstu dalších oblastí, které je potřeba do systému integrovat, dochází přímo úměrně i k nárůstu mapových podkladů a vrstev, které musí umět systém vizualizovat. Ať už se jedná o vrstvy, které obsahují čistě data sloužící k monitorování aktuální situace v dané oblasti, nebo o vrstvy, které poskytují množinu dat, nad kterými lze vykonávat analýzy a predikce, až po vrstvy, které slouží k výsledné vizualizaci provedeného výpočtu analýzy.

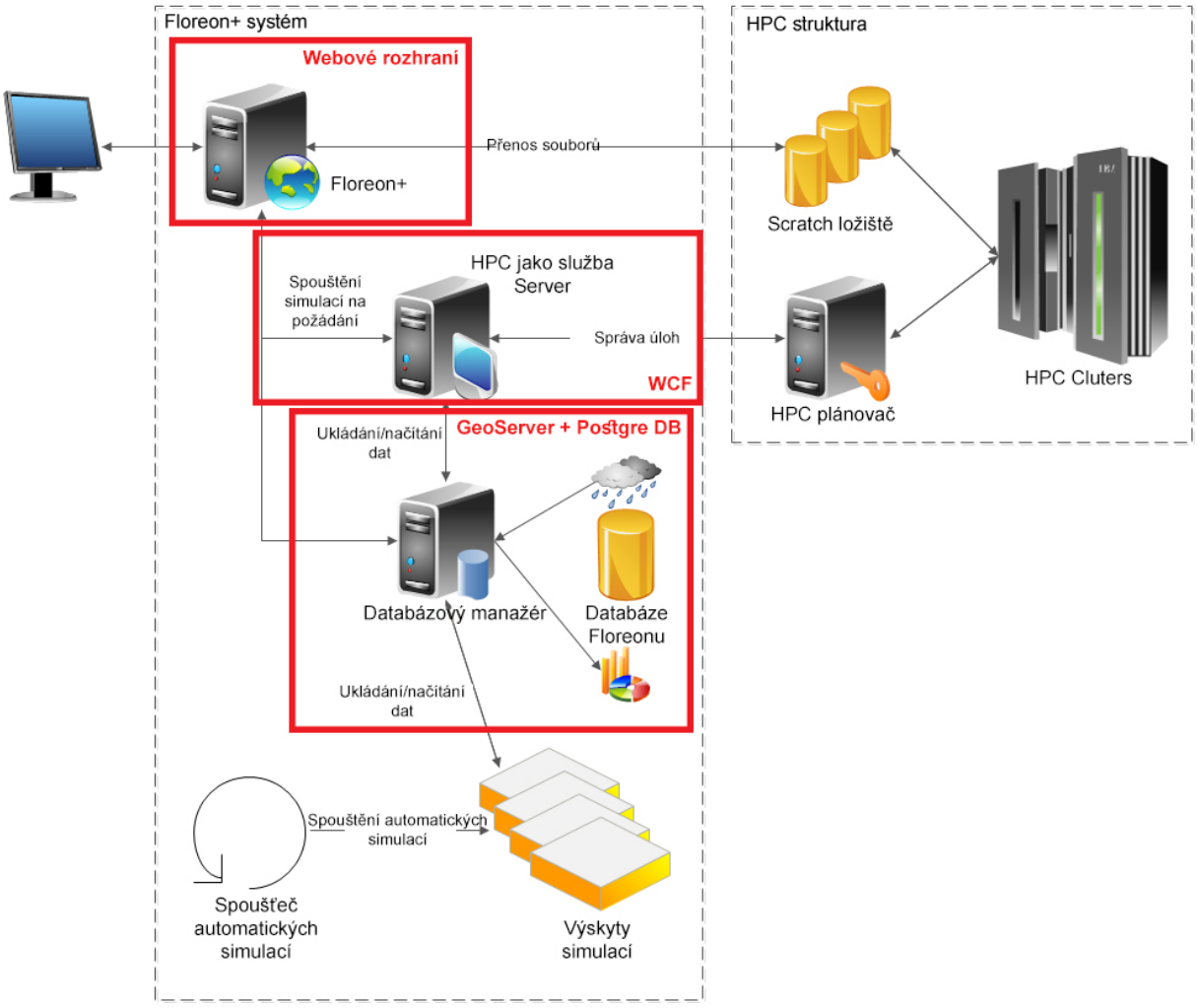
V původní verzi systému (dále označována jako Floreon+ 2.0) bylo pro každé takovéto rozšíření potřeba udělat zásah hned na několika místech ve zdrojovém kódu. Posléze, ačkoliv se nová mapová vrstva přidala „pevně“ v kódu, mohla nastat situace, kdy nebyla vůbec zobrazena množině uživatelů, neboť tato skupina neměla pro přístup k této vrstvě dostatečná oprávnění. Systém aktuálně definuje skupiny oprávnění od anonymních uživatelů bez přihlášení, až po roli „super administrátora“, která má přístup například k právě vyvíjeným analýzám a predikcím.

Cílem této práce bylo vyvinout funkcionalitu systému, která umožní přidávání těchto mapových vrstev výrazně jednodušším způsobem. Vzhledem k faktu, že práva i mapové vrstvy jsou definovány na úrovni serveru pro sdílení „geoprostorových“ dat (dále jen „GeoServer“, vizte kapitolu 3.2), lze vycházet při skladbě mapové kompozice jednotlivého uživatele, právě z nastavení tohoto serveru. Zodpovědnost obsahu se tedy po implementaci přesune z „front-endového“ kódu (kód tvořícího rozhraní, které využívá uživatel systému) na „back-end“ aplikace (kód logiky systému, většinou na serverové části), kde podle daných oprávnění uživatele se vygeneruje příslušná informace o mapových podkladech a vrstvách, která se posléze využije v uživatelském rozhraní k nadefinování mapového podkladu a jeho základních funkcionalit. Výsledná mapová kompozice je sestavena pouze z mapových vrstev, ke kterým má daný uživatel práva a se kterými může pracovat. Nová verze systému je nadále označována jako Floreon+ 3.0.

Obrázek 1: Logo projektu Floreon+

2 Architektura systemu

Architekturu systému Floreon+ lze rozdělit na dva základní celky a to „front-end“ (uživatelské rozhraní, „client-side“) a „back-end“ (tzv. „server-side“). Celou architekturu můžete vidět na obrázku číslo 2. Stranu uživatelského rozhraní reprezentuje webový klient, využívající technologie HTML, CSS a JavaScript (vizte kapitoly 3.1.1, 3.1.2 a 3.1.3). Přes toto webové rozhraní se klient může nově registrovat, opakovaně přihlašovat, upravovat si prostředí a celkově využívat funkcionalit systému - jako např. monitorování situací nad datovými vrstvami v reálném čase, nebo vytváření simulací a predikcí pomocí nástrojů, které systém poskytuje. Každý datový vstup do webového klienta a stejně tak výstup je prováděn pomocí volání služeb z „back-endového“ řešení.



Obrázek 2: Architektura systému Floreon+

„Back-end“ systému se skládá z GeoServeru a komunikačních modulů implementovaných pomocí WCF (vizte kapitoly 3.2 a 3.3). Hlavní funkcionalitou GeoServeru je vytváření mapo-

vých “tilů” (mapových obrazových čtverců) z datových zdrojů. GeoServer spravuje oprávnění jednotlivých uživatelů, nebo skupin uživatelů. Výstupem z GeoServeru je mapový podklad pro dotazovanou mapovou vrstvu ve formátu WMS nebo WFS (vizte kapitoly 3.2.1 a 3.2.2. V systému Floreon+ 2.0 je ve webovém klientovi napevno v kódu definovaná mapová vrstva, v rámci které dochází k volání příslušných API GeoServeru, přijímání vygenerovaných mapových „tilů“ a jejich vizualizace na webovém rozhraní. Webový klient, ačkoliv zkontroluje práva pro daného uživatele na GeoServeru, tyto vrstvy načítá pokaždé, dotazuje se na ně a posléze je skrývá nebo ponechává zobrazené podle uživatelských práv. Přidání nové mapové vrstvy tak čítá mnoho úprav - od definování mapové vrstvy v GeoServeru, přes vytvoření pevné instance této vrstvy ve webovém klientovi, až po kontrolu zda daný přihlášený uživatel má vůbec právo na to, danou vrstvu používat.

Ve systému Floreon+ 3.0 byla část procesu přidávání mapové vrstvy zautomatizována - zůstala potřeba definování vrstvy na úrovni GeoServeru a přiřazení práv skupině uživatelů k této vrstvě. Webový klient zbylou část provede automaticky - při připojení uživatele jej systém identifikuje a vyžádá si soubor, který definuje ke kterým vrstvám může daný uživatel přistupovat. Podle tohoto údaje vygeneruje mapovou kompozici a tudíž nemůže dojít k přidání mapové vrstvy, kterou by posléze uživatel nemohl používat. Další částí „back-endového“ řešení je WCF služba poskytující API pro podporu uživatelských činností.

Je nutno poznamenat, že architektura systému se nezměnila během úprav na dynamickou verzi mapové kompozice, hlavně z toho důvodu, že jedním z požadavků bylo minimalizovat rozsah změn ve stávajícím řešení. Hlavní změnou byla úprava webového rozhraní a způsobu komunikace s API GeoServeru. WCF taktéž zajišťuje služby jako autorizace a autentizace, dále pak například úkony „personalizace“ prostředí uživatele nebo vytváření a spouštění simulací a predikcí na superpočítači.

3 Použité technologie

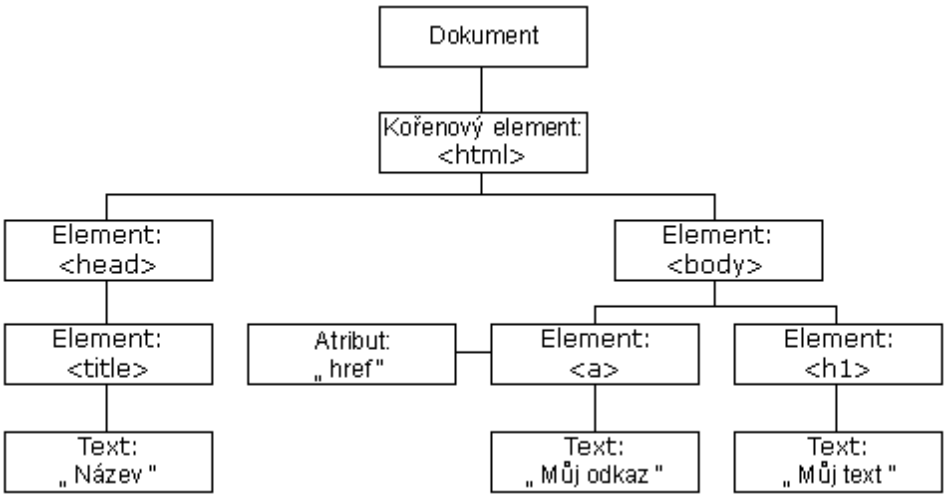
Následující kapitola popisuje jednotlivé dílčí technologie využité v systému Floreon+. Popis se vždy skládá z obecné stručné charakteristiky technologie a posléze způsobu využití v systému Floreon+.

3.1 Front-end aplikace

„Front-end“, nebo-li webový klient v našem případě, je postaven hlavně na technologiích HTML, CSS a JavaScript. V projektu se dodržuje izolace CSS a JavaScriptových kódů do jednotlivých samostatných souborů, odpovídající množině funkcionalit nebo vlastnosti webového rozhraní. Každá z využitých technologií je podrobněji popsána v následujících sekcích.

3.1.1 HTML

HyperText Markup Language (zkr. HTML) [1] je značkovací jazyk používaný pro tvorbu webových stránek. Struktura zápisu vychází z notace XML. Značky, tzv. „tagy“ spolu s jejich vlastnostmi („atributy“) tvoří tzv. elementy. Při načítání webové stránky dochází k vytváření DOM objektů z těchto elementů (příklad skladby HTML dokumentu pomocí DOM elementů je zachycen na obrázku číslo 3) a posléze se s těmito DOM objekty již dá zacházet jako s objekty, které známé z jiných programovacích jazyků jako jsou Java nebo C#. V našem případě k manipulaci s těmito objekty budeme používat JavaScript.



Obrázek 3: DOM struktura HTML dokumentu (zdroj: www.w3schools.com)

Pomocí jazyka HTML lze tedy stanovit strukturu webové stránky. Lze definovat oblasti, tabulky, texty, nápisy apod. typické části webové stránky. Pro propojení s kaskádovými styly

[1] https://www.w3schools.com/html/

(CSS) je nutno jednotlivé „tagy“ HTML dokumentu označovat identifikátory nebo třídami. Logicky jistě pochopíme, že v případě identifikátoru se musí jednat o jedinečné označení v rámci celého HTML dokumentu nebo případně celého webového projektu, zatímco vlastnosti dané třídou můžeme přiřadit hned řadě objektů. Pokud bychom v dokumentu přiřadili více objektům stejný identifikátor, webový prohlížeč by správně přiřadil definované vlastnosti všem objektům vlastnícím daný identifikátor, avšak webová stránka by se stala nevalidní. Problém by nastal v případě, kdy bychom chtěli jeden konkrétní prvek identifikovat pomocí JavaScriptu a například na něj navázat funkcionalitu. V takovou chvíli by nám totiž „selektor“ (jQuery anotace k výběru DOM elementu - vizte kapitolu 3.1.3.1) nevracel pouze jeden chtěný objekt, nýbrž pole všech objektů s daným identifikátorem. Ve systému Floreon+ 2.0 je prakticky kompletní webová stránka popsána explicitně právě v dokumentu HTML. Přesněji řečeno, každý jednotlivý element, který se na stránce může objevit je pevně definován v HTML dokumentu ačkoliv nikdy nedojde k jeho zobrazení.

Verze Floreon+ 3.0 je pochopitelně taktéž strukturovaná v jazyce HTML, avšak s jedním zásadním rozdílem souvisejícím právě s dynamickým přístupem k datům. Ve verzi Floreon+ 3.0 jsou již v struktuře HTML definovány explicitně pouze ty části webového rozhraní, které budou zobrazeny vždy a všem uživatelům. Zpravidla se jedná o hlavní rozvržení stránky (menu, časová osa, menu pro mapové podklady, stavová lišta, apod.). Elementy, které jsou závislé na právech uživatelů (např. formuláře pro zadávání analýz) vznikají až ve chvíli, kdy jsou reálně potřeba - tzn. pokud se na webovém rozhraní přihlásí uživatel, který nemá právo vytvářet simulace, nevygeneruje se mu ani v rozhraní DOM objekt reprezentující daný vstupní formulář. Zatímco ve verzi Floreon+ 2.0 by ve struktuře dokumentu formulář existoval, ale byl pouze podle práv uživatele skrytý, tak ve verzi Floreon+ 3.0 takovýto formulář ve zdrojovém kódu stránky ani nelze dohledat. Principy vytváření, upravování a mazání DOM elementů za běhu webové aplikace popisuje blíže kapitola 5.

3.1.2 CSS

Cascading Style Sheets (zkr. CSS) [2] je jazyk sloužící pro popis zobrazení jednotlivých elementů popsaných v jazycích HTML, XHTML nebo XML. Hlavním smyslem a taky důvodem vzniku tohoto jazyka je možnost oddělit vzhled dokumentu od jeho struktury a obsahu.

Syntaxe zápisu CSS stylu je velmi jednoduchá - skládá se z „selektoru“ a bloku deklarací. „Selektorem“ může být obecně HTML element (pak definované vlastnosti získají všechny dané elementy v HTML dokumentu, který používá daný CSS soubor), element označený jedinečným identifikátorem (posléze je vlastnost aplikována pouze na element nesoucí daný identifikátor) nebo na třídu (definované vlastnosti vlastní všechny elementy označené danou třídou). Pomocí CSS však nadále lze definovat i „primitivní“ chování webového dokumentu - jako je například chování elementu po jeho přejetí myší, apod. Méně šikovnou variantou výběru elementů je

[2] https://www.w3schools.com/css/

průchod dokumentu způsobem podobnému XPath (metoda průchodu XML dokumentu), tedy cyklickým „vnořováním“ se do jednotlivých elementů, až k elementu jemuž chceme přiřadit vlastnost. Tento způsob není příliš vhodný, neboť velmi snadno může dojít při modifikaci HTML dokumentu ke změně takto sestaveného selektoru a tudíž i narušení správného mapování vlastností na element.

Floreon+ 2.0 i 3.0 využívají externí CSS soubor, izolovaný od kódu HTML. Z toho důvodu, že verze 3.0 vznikla standardně odkloněním verze 2.0, dochází pouze k malým rozdílům v tomto souboru. Ve verzi 3.0 se však využívá dynamického vytváření DOM elementů a tak dochází k případům, kde v některých případech je nutnost vzhled definovat přímo při vytváření objektu. Také ovšem ve verzi 3.0 existují pozůstatky definovaných vzhledů z předchozí verze, které se nevyužívají. V tomto směru je do budoucnosti nutná „refaktorizace“ CSS souboru.

3.1.3 JavaScript

JavaScript [3] je multiplatformní, objektově orientovaný skriptovací jazyk, dnes velmi hojně využívaný k vytváření interakcí webových stránek. Syntaxe jazyka JavaScript patří do rodiny jazyků C/C++/Java. JavaScript je typickým „client-side“ jazykem, tzn. že se vykonává až po stažení webové stránky ke klientovi, takže se běh aplikace psané v JavaScriptu vykonává na straně uživatele. Dalším ze základních rysů JavaScriptu je dynamické typování - programátor nemusí explicitně definovat datový typ proměnných, typ se přiřadí podle přiřazené hodnoty do dané proměnné. Pomocí jazyka JavaScript lze dynamicky vytvářet DOM elementy webové stránky - teoreticky je možné celý web sestavit z DOM elementů, které se inicializují při spuštění stránky. Můžeme tak pomocí JavaScriptu definovat strukturu jednotlivých DOM elementů a také jim přiřadit vlastnosti pomocí CSS (příklad takto vytvořeného DOM elementu můžete vidět na obrázku 4). Tuto vlastnost je velmi důležité zmínit v souvislosti s webovým rozhraním systému Floreon+. Floreon+ 2.0 má v podstatě celou svou strukturu definovanou v souboru *index.html* a vlastnosti objektů jsou definovány v souborech kaskádových stylů. V systému Floreon+ 3.0 se začíná využívat onen druhý způsob vytváření DOM objektů a to právě z důvodu dynamického přístupu celé aplikace.

JavaScript je tedy hlavní jazyk pro webové rozhraní systému Floreon+. Pomocí JavaScriptu se definují jazykové mutace, chování času (na rozhraní se dá simulovat posun času, nebo vizualizovat historické události), způsoby „personalizace“ systému jednotlivým uživatelům a také funkcionality spjaté s mapovou kompozicí.

3.1.3.1 Framework jQuery Webové rozhraní systému Floreon+ využívá nadstavbu jazyka JavaScript - framework jQuery [4]. jQuery je JavaScriptová (vizte výše) knihovna s širokou podporou prohlížečů, která klade důraz na interakci mezi JavaScriptem a HTML kódovacím jazykem. Tato knihovna je vydávána pod volně šiřitelnou licencí MIT. jQuery zkracuje JavaScriptový

[3] https://www.w3schools.com/js/
[4] https://www.w3schools.com/jquery/

Obrázek 4: Dynamicky vytvořený DOM element

zápis kódu, ulehčuje jeho psaní, poskytuje snadnou a rychlou pomoc při hlídaní událostí nebo například vytváření animací. Taktéž umožňuje značné ulehčení využitím Ajaxu.

3.1.3.2 Ajax Další funkcionalitou JavaScriptu, resp. jQuery je tzv. Asynchronous JavaScript and XML (zkr. Ajax)[5], který umožňuje měnit obsah stránek bez nutnosti kompletního znovu načtení stránky. Jedná se o asynchronní zpracování webových požadavků - např. získávání dat z „back-endu“ aplikace, nebo naopak odesílání dat k jejich zpracování nebo uložení. Název typicky evokuje přenos dat ve formátu XML, který byl dříve nejvíce v tomto kontextu využíván, avšak dnes se začíná mnohem více využívat formát JSON, který je v podstatě slovníkovým výčtem dat. Každý údaj má vlastní klíč a hodnotu, ovšem lze v JSON objektu zachytit i například pole - tudíž k jednomu klíči můžeme přiřadit další celý „podslovník“. Samotné volání má velmi jednoduchou anotaci - definuje se URL služba, kterou chceme volat, typ volání (např. GET, POST,...), formát v jakém data posíláme a data samotná. K požadavku se taktéž připojí „události“ pro případ úspěchu či neúspěchu (mohou se definovat i další) volání služby.

3.1.4 OpenLayers framework

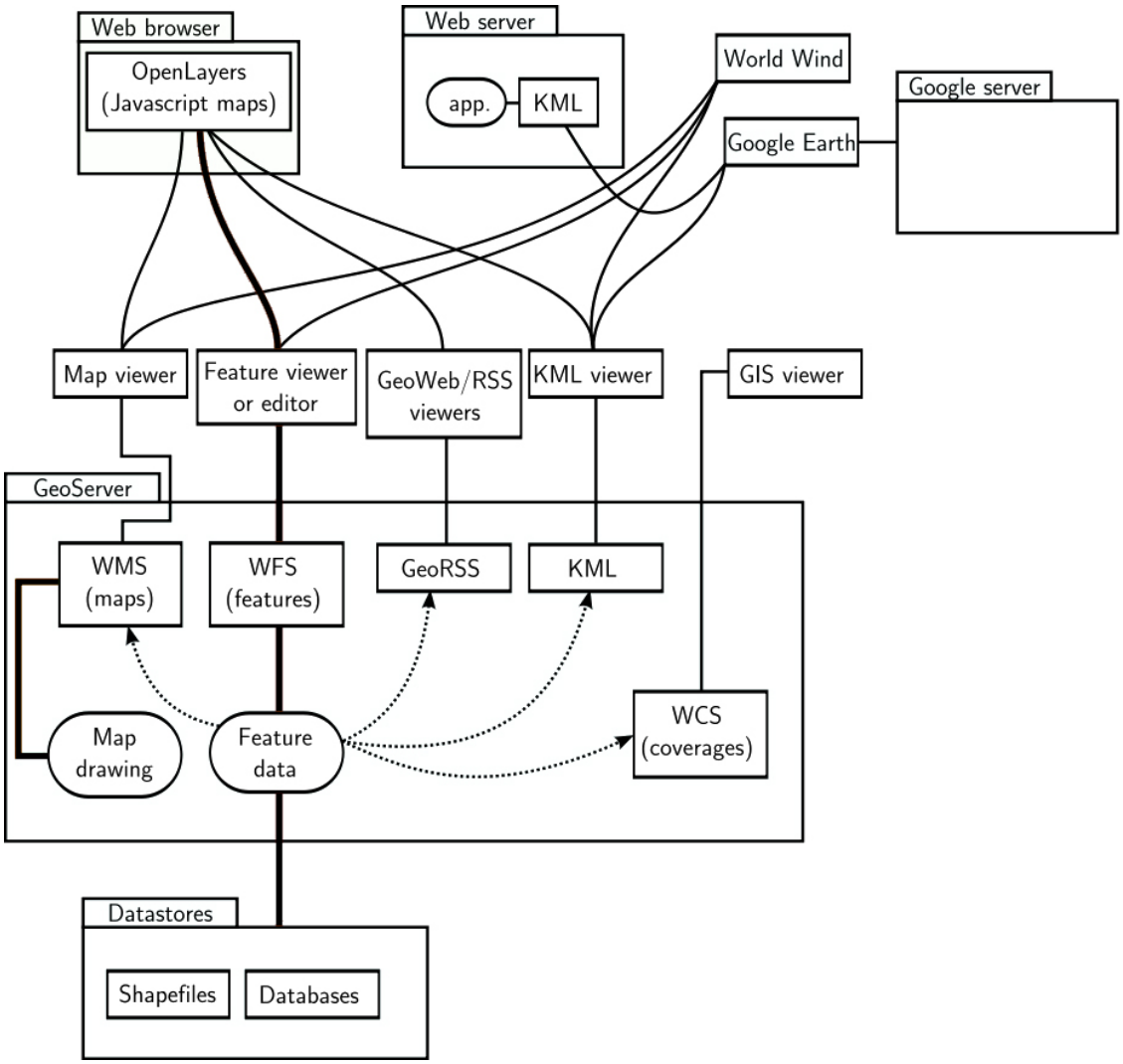
OpenLayers je další JavaScriptový framework určený k vytváření webových mapových aplikací. OpenLayers poskytuje funkce pomocí kterých lze napojit webové rozhraní na zdroj mapových podkladů (v našem případě GeoServer) a posléze z poskytnutých grafických dat seskládat mapovou kompozici (typy dat a jejich napojení je zachyceno v diagramu na obrázku **??**). OpenLayers taktéž poskytuje posléze širokou škálu funkcionalit jak s mapovou kompozicí pracovat. Kromě tedy vizualizace geografických obrázkových dat (vizte kapitolu 3.2.1), umí také zobrazovat data vektorová (nepracuje přímo s rastrovým obrázkem, nýbrž s matematicky definovaným objektem, který OpenLayers umí na mapě vhodně vizualizovat) tzv. WFS (vizte kapitolu 3.2.2) nebo

[5] https://www.w3schools.com/xml/ajax_intro.asp

přidávat do mapy elementy z dalších externích zdrojů (např. význačné body a data ke každému z těchto bodů). Framework byl vytvořen společnosti MetaCarta a publikován jako tzv. „open-source“ software v roce 2006.

Aktuální produkční verze systému Floreon+ využívá framework OpenLayers verze 2, avšak ve vývoji dnes existují dvě hlavní vývojové větve a to Floreon+ rozhraní s OpenLayers 2, kde se implementovaly další funkcionality dle potřeb systému a Floreon+ rozhraní s OpenLayers verzi 3 [5], kde přechod na novou verzi bylo pouze logické využití příležitosti při přepisu systému na dynamický přístup při procesu vytváření mapové kompozice.

Je nutností také zmínit skutečnost, že OpenLayers verze 3 není zpětně kompatibilní se svým předchůdcem [1], verzí 2, což ve finále znamená, že přechod systému, který využívá OpenLayers 2 na OpenLayers 3, je obzvláště složitý a komplexní, neboť obrovská většina funkcionalit navázaných na verzi frameworku musí být přepsáno nebo implementováno znovu.



Obrázek 5: OpenLayers a typy dat

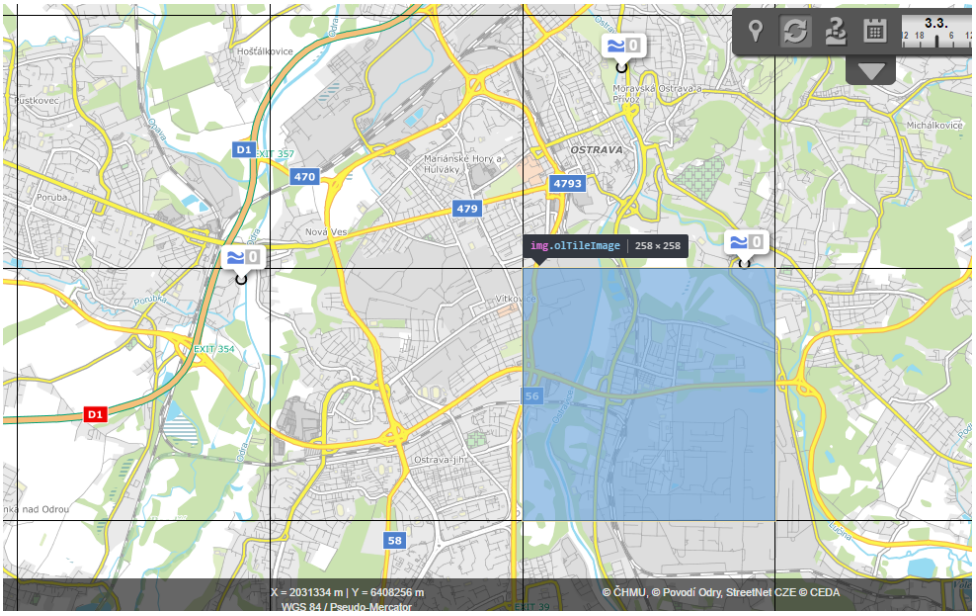
3.2 GeoServer

GeoServer [4] je „open-source“ server, který umožňuje sdílet, zpracovávat a upravovat data, která se označují jako „geoprostorová“. Projekt je zaměřen hlavně na interoperabilitu a publikuje data s využitím otevřených standardů, které definuje Open Geospatial Consortium (OGC) [3]. GeoServer implementuje standardy jako WMS, WFS, WCS a další [2], a umožňuje vizualizovat data mnoha různými způsoby a formáty. Již zmíněný JavaScriptový framework OpenLayers je součástí základní instalace GeoServeru, kde slouží jako nástroj pro vizualizaci dat. GeoServer využívá PostGIS nadstavby pro objektově-relační databázový systém PostgreSQL (obě technologie naleznete popsané v kapitole 3.2.4), který přidává podporu pro tzv. „geoprvky“. PostGIS implementuje specifikaci „Simple features for SQL“ konsorcia OGC. PostGIS tedy zahrnuje podporu pro geometrické útvary (lomené čáry, polygony, ...), funkce pro průniky geoprvků, funkce pro výpočty vzdáleností, výměr ploch a obvodů v 3D prostoru. PostGIS taktéž vychází pod „open-source“ GNU/GPL licencí. V systému Floreon+ je GeoServer, už ze své podstaty, součástí „back-endu“ aplikace, resp. bychom o GeoServeru měli mluvit jako o množině instancí GeoServerů, neboť v systému není pouze jedna instance, kterou by využíval každý klient, nýbrž hned několik instancí, které operují nad stejnými daty a jednotliví klienti při připojení na webové rozhraní jsou pomocí proxy připojeni k jedné z těchto instancí. Důvod této implementace je snížení zátěže (množství požadavků) na jednotlivou instanci GeoServeru.

3.2.1 WMS standard

Web Map Service (zkr. WMS) je standard vyvinutý a udržovaný konsorciem OGC. Služba umožňuje sdílet geografické informace ve formátu rastrových map. Výsledkem požadavku jsou tak obrazová data ve známých formátech (např. JPEG, PNG, TIFF, ...). Množina tématicky stejných geografických dat tak tvoří mapovou „vrstvu“ a překryv takovýchto vrstev tvoří „mapovou kompozici“. Příkladem jedné WMS vrstvy může být například samotná silniční síť České republiky, a jako kompozici si lze představit kombinaci silniční sítě a mapovou vrstvu obsahující vytížení jednotlivých silničních úseků dle času. Kombinací těchto dvou samostatných mapových vrstev vzniká mapová kompozice, na které lze pozorovat aktuální (resp. pro zadané časové období) vytížení silnic a dálnic. Obrazová data se obecně dají zobrazovat dvěma způsoby - umístění celého obrázku do mapy pomocí „bounding boxu“ - umístění do oblasti mapy pomocí souřadnic, nebo zobrazení v mapové kompozici jako tzv. „tily“. „Tile“ je označení pro čtvercovou část mapové mozaiky, kde jednotlivé „tily“ umístěny vedle sebe tvoří souvislou mapovou vrstvu - vizte obrázek 6.

Mezi nejdůležitější typy dotazů na GeoServer v souvilosti s WMS patří *GetMap*, který slouží k získaní mapové vrstvy dle zadaných atributů (formát, verze, „bouding box“, ...). Dalším dotazem je *GetCapabilities*, což je dotaz o kterém budeme v této práci asi nejvíce hovořit, neboť tímto dotazem uživatel získává informace o dané mapové vrstvě a taktéž se pomocí *capabilities* dozvídá jak s danou vrstvou pracovat. Třetím nejdůležitějším dotazem je *GetFeatureInfo*. Ačkoliv



Obrázek 6: Příklad tile částí mapy v systému Floreon+

WMS služba vrací mapové data ve formátu rastrových obrázků, lze pořád aplikovat dotaz na objekty obsažené v mapě. Co to znamená v praxi? Řekněme, že máme mapovou vrstvu, která obsahuje všechny restaurace v ČR. I přes skutečnost, že restaurace (značena například symbolem příboru) je součástí rastrového obrázku, můžeme získat o této restauraci informace po například kliknutí na ikonku příslušné restaurace. To je možné díky schopnosti GeoServeru uchovávat data k daným mapovým objektům a vracet o nich informace na základě jejich umístění. Ačkoliv jsme tedy na mapě neklikli na samostatný element/objekt, ale na část obrázku nebo dokonce jen část obrázkového „tilu“, můžeme získat díky souřadnicím informace o daném elementu/objektu.

V systému Floreon+ 2.0 je WMS formát využíván hlavně na podkladové mapové vrstvy a na několik dalších funkčních mapových vrstev, nicméně také Floreon+ 2.0 využívá hojně formát WFS. Floreon+ 3.0 je aktuálně postaven jen na formátu WMS a to z toho důvodu, že v rámci přechodu z Floreon+ 2.0 na 3.0 došlo taktéž k implementaci WPS funkcí na „back-endu“ aplikace, kterými se nahradí funkce doposud implementované v uživatelském rozhraní nad vektorovými elementy/objekty.

3.2.2 WFS standard

Web Feature Service (zkr. WFS) je taktéž standard vyvinutý a udržovaný konsorciem OGC. Služba WFS umožňuje sdílení dat ve vektorovém formátu. Výsledným formátem dat jsou primárně geodata GML. Dalšími typickými formáty jsou např. JSON[6], GeoJSON[7] nebo CSV[8]. Geografická data služby WFS jsou geometrické útvary (bod, linie, plocha), které jsou vztaženy

[6] http://www.json.org/json-cz.html
[7] http://geojson.org/geojson-spec.html
[8] https://cs.wikipedia.org/wiki/CSV

k referenčnímu souřadnicovému systému - nejčastěji udávaném v datasetu EPSG[9]. Asi nejzásadnějším rozdílem oproti WMS je však následná práce z WFS daty - zatímco v případě WMS jsou data získaná od serveru finální (jedná se o obraz), v případě WFS mohou být data ještě mnohokrát modifikována, než se zobrazí v mapové kompozici. Pro konkrétnější představu - mějme body na mapě, které reprezentují, například opět, restaurace na území města Ostravy a polygon označující území části Ostravy-Poruby. Obě datové množiny pocházejí z dvou různých WFS dotazů. V momentě kdy klientská aplikace získá oboje data, může vykonat mezi těmito daty funkci průnik, a získat tak body se souřadnicemi všech restaurací na území městské části Ostrava-Poruba. Dalším krokem pak může být zobrazení takto vzniklého datového setu v mapové kompozici, nebo jakákoliv jiná další operace. Samotný dotaz směřovaný na server může obsahovat tzv. CQL filtr[10], pomocí kterého lze vybrat podmnožinu dat.

V systému Floreon+ 2.0 se například WFS služba využívala pro zobrazování měřících stanic, kde právě WFS dotaz vracel souřadnice jednotlivých stanic a příslušné informace v daném čase. Při načítaní webového rozhraní se pak na tyto souřadnice umístily SVG obrázky, které obsahovaly i příslušné data k dané měřící stanici (povodňový stupeň). Ve verzi Floreon+ 3.0 je vrstva stanic používána jako WMS standard a SVG obrázky tak nahrazují symboly přímo ve WMS vrstvě. Nejvíce se tedy WFS standard využívá pro vizualizaci výsledku analýz. Celkově však v systému Floreon+ 3.0 došlo k snížení užívání tohoto standardu, a to především z důvodu omezení maximálního počtu WFS elementů (bodů, křivek), které umí OpenLayers načíst. Jinak řečeno, OpenLayers má ve výchozím nastavení omezení pro načítání prvků z WFS standardu, a to na přesně jeden tisíc elementů. Omezení se dá změnit, nicméně podle dokumentace OpenLayers [1] se takováto změna nedoporučuje z výkonnostních důvodu. Omezení se projevilo v případě implementace analýzy pro dálniční a silniční síť ČR, kde jen na území Prahy je výskyt zhruba 322 tisíc elementů označujících buď uzel, nebo hranu. Toto omezení přispělo velkou mírou k soustředění mapové kompozice do standardu WMS.

3.2.3 Capabilities

Capabilities jsou data v XML formátu, jež jsou odpovědí na volání služby *GetCapabilities* WMS nebo WFS GeoServeru. Data v odpovědi na takové volání obsahují výčet všech operací, služeb a celkově schopností, které server podporuje. Typické volání se skládá ze 3 povinných parametrů - *service*, *version* a *request*. *Service* označuje, zda se dotazujeme na WMS nebo WFS standard, resp. WMS nebo WFS službu (jedna instance GeoServeru obsluhuje jak WMS tak WFS dotazy). Parametr *version* označuje verzi *capabilities* na kterou se chceme dotazovat, a parametr *request* určuje o jaký typ požadavku se jedná - v našem případě bude hodnota jak pro WMS tak i WFS server *GetCapabilities*. Pokud by některý z parametrů byl v požadavku vynechán, GeoServer jej doplní výchozí hodnotou - nicméně standardním přístupem je tyto parametry explicitně do požadavku zahrnout. Nepovinnými parametry jsou pak *namespace* a *format*, kde *namespace*

[9] https://www.epsg-registry.org/

[10] http://docs.geoserver.org/stable/en/user/data/app-schema/cql-functions.html

umožňuje dotazovat se na *capabilities* jen pro určitý jmenný prostor vrstev (jmenný prostor je množina vrstev se společným základním rysem - např. v systému Floreon+ obsahuje jmenný prostor *basemaps* jen množinu těch vrstev, které mohou být jen podkladové, tedy ty vrstvy, které vždy budou překrývány jinými vrstvami, a zároveň samy nikdy žádnou jinou než *base* vrstvu nepřekryjí) a parametr *format* umožňuje vyžádat si *capabilities* data v jiném formátu než výchozím XML. Vzhledem k tomu, že na úrovni GeoServeru se také definují uživatelská oprávnění (resp. oprávnění pro skupiny uživatelů - jejich role), lze dotaz *GetCapabilities* rozšířit o další parametr tzv. *authKey*. *AuthKey* nebo-li také pojem *token* je označení unikátního identifikátoru nějakého objektu. V našem případě tím daným objektem, který je třeba identifikovat je uživatel. Znamená to tedy, že každý uživatel (minimálně registrovaný) systému Floreon+, má svůj jedinečný identifikátor. Jedinou výjimku tvoří tzv. anonymní uživatel, což je návštěvník webového rozhraní, který zde nemá vytvořený vlastní účet. Každý takovýto uživatel, s jedinečným identifikátorem, patří do skupiny uživatelů, kteří mají přiřazenou roli v systému. Role můžou být od anonymního uživatele až po super administrátora. Jakou roli danému uživateli přiřadit určuje uživatelův jedinečný identifikátor, který patří do definované skupiny práv. Jestliže tedy uživatelův identifikátor bude ve skupině práv, která má nadefinovány přístupy a možnosti super administrátora - dostane tento uživatel práva super administrátora. V *GetCapabilities* požadavku hraje tento uživatelský identifikátor významnou roli. Podle toho, na jakou množinu vrstev má daný uživatel právo, takovou množinu i požadavek vrátí. Reálně to posléze znamená, že pokud se klient dotáže na server z role nepřihlášeného uživatele, vrátí se mu omezené množství vrstev, které může sledovat ve své mapové kompozici. Ve chvíli kdy se uživatel přihlásí, změní se jeho role, a požadavek *GetCapabilities* s *authKey* parametrem obsahujícím unikátní identifikátor přihlášeného uživatele vrátí rozdílné množství vrstev. Tato funkcionalita je klíčová pro tuto práci. Systém Floreon+ 2.0 totiž napevno definuje veškeré mapové vrstvy, doptává se na ně GeoServeru, ale ve finále se uživateli nemusí vůbec zobrazit, neboť fakt, zda na tyto vrstvy uživatel má práva, kontroluje až posléze a pouze je uživateli skrývá. Systém Floreon+ 3.0 využívá plného potenciálu *capabilities* a sestavuje uživateli mapovou kompozici přesně podle toho, jaké oprávnění daný uživatel má. Více vizte v kapitole 5.

3.2.4 PostgreSQL a PostGIS

PostgreSQL[11] databáze je „open-source“ objektově-relační databázový systém. Je plně v souladu s ACID přístupem a plně podporuje veškeré klasické prvky, které bychom u RDBMS očekávali - cizí klíče, nadhledy, spojení, uložené procedury atd. Funkce PostgreSQL mohou být psány např. v jazyce PL/pgSQL, Java, Python nebo i například ve statistickém jazyce R. PostgreSQL je vyvíjena skupinou „PostgreSQL Global Development Group“, což je skupina lidí skládající se z jednotlivých skupin dílčích společností a jednotlivců. V systému Floreon+ je PostgreSQL databáze využívána s nadstavbou PostGIS[12] k uchovávání prostorových dat. Kromě „statických“

[11] https://www.postgresql.org/
[12] http://postgis.net/

dat, které jsou GeoServerem reprezentovány jako pevné vrstvy (např. mapová vrstva povodí), se do prostorové databáze ukládají vypočtené analýzy a predikce. Každá takto uložená analýza má minimálně svůj unikátní identifikátor a časovou stopu, kdy vznikla. Analýza vzniká jejím vytvořením uživatelem systému Floreon+, který má dostatečná oprávnění k vytváření analýz a predikcí. Tento uživatel na webovém rozhraní zvolí typ analýzy a vyplní veškeré relevantní informace a data nutné k úspěšnému výpočtu analýzy nebo predikce. Při odeslání a uložení analýzy se dějí v zásadě dvě základní události - uložení dat o analýze do MSSQL databáze (vizte kapitolu 3.3.0.1), aby se analýza mohla přiřadit danému uživateli (více v kapitole 3.3), a odeslání výpočtu do fronty na superpočítač Salomon nebo Anselm. Jakmile je analýza vypočtena superpočítačem, jsou výsledná data analýzy uložena do PostGISové prostorové databáze. Uživatel je posléze upozorněn, že došlo k výpočtu analýzy a analýza je připravena k vizualizaci na webovém rozhraní. Uživatel ve svém účtu zvolí zobrazení vypočtení vypočtené analýzy, čímž pošle požadavek na GeoServer, který z prostorových dat výsledku analýzy sestaví data ve formátu WFS (nebo WMS) a vrátí zpět klientovi, kde je výsledek zobrazen v mapové kompozici.

3.3 WCF

Windows Communication Foundation (zkráceně WCF)[13] je framework technologie .NET, který se skládá z sady knihoven a API, které zajišťují komunikaci mezi aplikacemi a umožňují tak vytvářet servisně orientované aplikace. Služba WCF vystavuje kolekci koncových bodů, kde každý jeden koncový bod představuje bránu, přes kterou komunikuje s okolím. Jakmile je služba WCF spuštěna, začne naslouchat svému okolí zda nějaký klient nevyžaduje komunikaci. WCF služba má mnoho vlastností, kde mezi ty nejdůležitější patří zabezpečení (šifrování dle známých standardů, jako např. SSL), rozšiřitelnost (snadné rozšíření o řadu komponent) a pro nás hlavně podpora AJAX a REST. Technologie AJAX byla popsána v kapitole JavaScript a je nejpodstatnější pro webové rozhraní systému Floreon+ v souvislosti se službou WCF. WCF služba umí zpracovat REST formát požadavků, ale taktéž formát SOAP bez dodatečných SOAP anotací, a také non-XML formáty, jako např. JSON formát. V systému Floreon+ zastávají moduly implementované pomocí WCF služby pomyslnou druhou část „back-endu“ aplikace. Proto nadále pod pojmem WCF služba budeme mluvit o těchto implementovaných modulech. WCF službu můžeme označit jako pomyslnou část „back-endu“ protože komunikuje i s GeoServerem, avšak z pohledu webového klienta se liší volání služeb GeoServeru a WCF. Zatímco GeoServer volání se používá hlavně k operacím spojeným s mapovou kompozicí, WCF služba se využívá hlavně k „personalizaci“ systému jednotlivým uživatelům a poskytování individuálních funkcionalit a dat. Celkově WCF služba zastává v systému Floreon+ tyto funkcionality:

- Registrace nového uživatele do systému
- Přihlašování do systému

[13] https://msdn.microsoft.com/en-us/library/dd456779(v=vs.110).aspx

- Odhlašování ze systému
- „Personalizace prostředí“ - individuální preferovaná nastavení jako např. preferovaný souřadnicový systém
- Mapové „bookmarks“ - uložení vybrané oblasti a konfigurace mapové kompozice (zobrazené mapové vrstvy)
- Kontrola uživatelských oprávnění
- Správa analýz a predikcí

  Vytváření analýz a predikcí

  Spouštění analýz a predikcí na clusteru superpočítače

  Kontrolování stavu čekajících analýz

  Předávání dat pro vizualizaci klientovi

  Mazání a uložených analýz a predikcí

Každý uživatel, který navštíví webové rozhraní systému Floreon+ nevědomky komunikuje s WCF službou. V momentě návštěvy dochází hned v prvním momentu k rozeznávání uživatele - zda má v *cookies* webového prohlížeče záznam o uživatelském identifikačním klíči, kterému ještě nevypršela validita. Pokud je takový *token* u uživatele nalezen, je zaslán dotaz na WCF, který uživatel takový „token“ vlastní. Služba posléze vrací zpět informace o uživateli a JavaScript v klientovi vyhodnotí příchozího uživatele jako přihlášeného. Pokud *token* nalezen není, bude uživatel vůči systému vystupovat jako anonymní návštěvník a to až do chvíle, kdy se uživatel registruje a získá tak vlastní identitu v rámci systému. V systému Floreon+ 3.0 se WCF využívá také na definování množin vrstev uživatele, nebo slouží k lokalizaci webového rozhraní. WCF využívá k ukládání dat MS-SQL databázi, která je nadále popsaná v následující kapitole.

3.3.0.1 MS-SQL databáze Microsoft SQL Server (MS-SQL)[14] je dalším typem relačního databázového systému. Tak jako ostatní RDBMS je postaven na jazyku SQL. MS-SQL databáze je vázána k „Transact SQL“ (zkr. T-SQL), což je SQL jazyk rozšířený o řadu proprietárních programovacích rozšíření standardního SQL jazyka. V systému Floreon+ je MS-SQL databáze používána z největší části pro účely ukládání dat o uživatelích. V rámci řešení Floreon+ 3.0 slouží MS-SQL databáze jako podpora pro vytvoření dynamického přístupu celého webového klienta. Konkrétním příkladem může být vytvoření tabulek v MS-SQL databázi, které slouží jako slovník pro lokalizaci webového klienta. Zatímco v systému Floreon+ 2.0 byly veškeré jazykové mutace udržovány přímo v klientovi, v systému Floreon+ 3.0 se načítají z databáze.

[14] https://cs.wikipedia.org/wiki/Microsoft_SQL_Server

4 Floreon+ 2.0

Tato kapitola se věnuje webovému rozhraní systému Floreon+ 2.0, jinak řečeno jak webové rozhraní a mapová kompozice fungovaly před započetím implementace úprav na Floreon+ 3.0. V první řadě je nutno říci, že systém Floreon+ je jedním z projektů vyvíjených vysokoškolským ústavem IT4Innovations, týmem „ADAS“. Prvním krokem k přechodu na Floreon+ 3.0 bylo vytvoření vývojové větve pro tento účel, neboť vývoj nových funkcionalit se nemohl zastavit z důvodu přechodu na novou verzi OpenLayers. Přechod na OpenLayers verze 3 se úzce pojí s zadáním této práce, avšak samotný přechod na novou verzi OpenLayers byl pouze logický krok v souvislosti s úkolem vytvoření dynamické kompozice mapových vrstev. Jinak řečeno, přechod na novou verzi OpenLayers se rozhodlo ve chvíli kdy vzniklo zadání této práce, neboť převedení systému na dynamickou verzi prakticky znamená přepsání aplikace znovu. Vzhledem k faktu, že OpenLayers verze 3 jsou zpětně nekompatibilní, znamenalo to prakticky implementovat veškeré části systému navázané na verzi OpenLayers znovu. Z toho důvodu se musely vývojové větvě odklonit. Existovala tedy v jednu chvíli vývojová větev Floronu 2.0 a 3.0. Ve verzi 2.0 se nadále vyvíjely úkoly, které měly pevně daný termín dokončení a ve verzi 3.0 vznikl prostor pro předělání systému na dynamickou kompozici. Floreon+ 2.0 tedy měl staticky nadefinované mapové vrstvy - to znamená, že napevno v kódu webového klienta byly definovány všechny mapové vrstvy, které bylo třeba přidat do mapové kompozice a tudíž přidání jakékoliv nové vrstvy znamenalo zásah programátora. Hlubšímu náhledu do mapové kompozice systému Floreon+ 2.0 se věnuje podkapitola 4.2.

4.1 OpenLayers 2

Floreon+ 2.0 využívá jako hlavní mapový framework OpenLayers 2. Poslední „release“ této verze (konkrétně verze 2.13.1) byl vypuštěn 9. července 2013. Od prvního „releasu“ verze OpenLayers 3 hlásají veškeré dokumentace OpenLayers 2, že se nacházíte v zastaralé verzi, která již nebude nadále udržována. Nejzásadnější rozdíl mezi OpenLayers 2 a OpenLayers 3 tvoří využití nejnovějších technologií ve verzi 3. Zatímco nová verze plně využívá možností technologie HTML5 a CSS3 (více v kapitole 5.1), OpenLayers 2 zaostává na předchozích verzích HTML a CSS, což je v mnoha ohledech velmi limitující.

4.2 Statická mapová kompozice

Po přechodu na Floreon+ 3.0 můžeme způsob skládání mapové kompozice z předchozí verze označit za statickou. Statickou z toho důvodu, že veškeré mapové vrstvy, které se v systému zobrazují, musejí být definovány v kódu klienta. V JavaScriptu webového rozhraní musí existovat část kódu pro každou jednotlivou vrstvu v systému. V takové definici pak musí být obsaženo, jak správně zavolat službu GeoServeru, formát v jakém chceme odpověď, zda daná mapová vrstva je podkladovou nebo není a nadefinovat jaké styly na mapovou vrstvu použít - při definici musí být

napevno zadán název stylu. Dále pak název mapové vrstvy v OpenLayers a název mapové vrstvy v GeoServeru atd. Pokud vrstva měla na sebe navázanou funkcionalitu dotazování (lze do mapové vrstvy kliknout a očekávat informace o elementu, na který byl klik proveden), bylo potřeba v kódu definovat událost kliknutí na danou mapovou vrstvu. Pokud kliknutí směřovalo na element zapnuté mapové vrstvy, bylo třeba definovat, kam se má webový klient dotazovat, aby získal příslušná data o elementu. Jakmile klient dostane data v odpovědi, musí se pevně definovat, jak se data zobrazí a co s těmito daty bude možno provádět nadále. Pokud by se například změnil název vrstvy v GeoServeru, mapová vrstva by se přestala zobrazovat. Respektive nejen, že by se nezobrazila, ale způsobila by na rozhraní prázdnou mapovou vrstvu, která z důvodu nedostupnosti požadovaných dat byla vyplněná prázdnými „tily“, které ve výchozím nastavení mají růžovou barvu. Proces přidání nové mapové vrstvy vypadal tedy následovně - do Postgre SQL databáze se nahrály prostorová data mapové vrstvy, posléze v GeoServeru se definovala nová vrstva a nastavily se její vlastnosti. Na straně klienta bylo třeba vytvořit novou definici mapové vrstvy - vytvořit objekt mapové vrstvy pro OpenLayers framework (příklad vytvoření objektu mapové vrstvy lze vidět v ukázce kódu ve výpisu 1).

```
var jsdi_events_2 = new OpenLayers.Layer.WMS("jsdi_events_2_time_10min",
    GeoServer_URL + "/rodos/ows",
     {
         layers: "rodos:jsdi_events_2",
         transparent: "true",
         format: "image/png",
         time: getUTCTimeForLayers(10),
         authkey: userToken,
     },{
         style: {
             pointRadius: 3,
             strokeWidth: 1,
             strokeColor: "black",
             fillColor: "red",
             cursor: "pointer",
         },
         visibility: false,
             displayInLayerSwitcher: false,
         group: "traffic",
         singleTile: true,
         geoNameLayer: "rodos:jsdi_events_2"
     });
```

Výpis 1: Definice mapové vrstvy JSDI

Takto vytvořený objekt se musel správně napojit na GeoServer tak, aby se objekt naplnil daty z GeoServeru. V ukázce kódu výpisu 1 je zachycená celá potřebná definice pro mapovou vrstvu dopravních informací. Pokud nová mapová vrstva patřila do již vytvořených skupin vrstev (Podkladové mapy, Floreon+ - mapové vrstvy relevantní se záplavami, Rodos - mapové vrstvy relevantní s dopravou, ...), muselo se definovat do které skupiny patří. Pokud ovšem se jednalo o vrstvu, která nespadala ani do jedné z těchto kategorií, nastala komplikace. Musela se nejprve definovat celá nová kategorie, která od chvíle vzniku existovala i v případě, že by neexistovala žádná mapová vrstva, která by do dané kategorie (skupiny vrstev) spadala. A až do takto vzniklé kategorie jsme mohli novou mapovou vrstvu přiřadit. Pokud mapová vrstva obsahovala pouze mapová data a jejich hodnota spočívala pouze v zobrazení v mapové kompozici, pak byl proces přidání nové mapové vrstvy dokončen. Vázala-li se však k mapové vrstvě funkcionalita dotazování na mapové elementy - například kliknutím na prvek ve vrstvě, musel se definovat pomocí funkcionalit OpenLayers příslušný „kontroler“, který po kliknutí na danou mapovou vrstvu provedl předem definovanou posloupnost úkonu, které s příslušnou vrstvou souvisely (například zobrazení relevantních dat daného elementu).

Pro přiblížení si uveďme jeden příklad, opět na již dříve zmiňované vrstvě, obsahující informace o umístění restauračních zařízení na území města Ostravy. Mapovou vrstvu nadefinujeme podobně jako v příkladu s JSDI vrstvou a přidáme vrstvu do mapy. Pokud všechny parametry jsou nastaveny správně, měla by se nově přidaná vrstva zobrazit v mapové kompozici. Ovšem zobrazí se pouze jako vizuální mapová vrstva, bez jakýchkoliv možností interakce. U takovéto vrstvy bychom však očekávali minimálně míru interakce v tom směru, že pokud uživatel provede kliknutí na element restaurace v mapové kompozici, tak se mu např. restaurace označí a zobrazí se název restaurace. Pokud v prostorové databázi tyto informace máme, tak GeoServer nám je umí také dále interpretovat. Avšak pokud není nadefinováno chování při kliknutí na daný element, nic se nestane. Takže v tomto ohledu, při přidání nové vrstvy je opět nutný zásah programátora. V klientské aplikaci (webové rozhraní) se musí nadefinovat tzv. „kontroler“. V „kontroleru“ se definuje, ke které vrstvě se akce vztahuje a jak se má uživatelské rozhraní zachovat, když k události kliku dojde. Pokud je „kontroler“ správně nastavený, dojde při kliknutí na element v dané vrstvě k vyvolání dotazu na GeoServer. GeoServer rozpozná, který element (případně které elementy) byl zasažený událostí kliknutí a relevantní informace jsou navráceny zpět klientovi v odpovědi na volání služby GeoServeru, opět, v předem definovaném podporovaném formátu (např. JSON). V klientské aplikaci pak musí mít definováno, jak získaná data zobrazit. Ve webovém rozhraní k těmto účelům slouží pravý vysouvací panel. Navrácena odpověď je tedy poskládána do HTML struktury a zobrazena v pravém panelu.

V systému Floreon+ 2.0 je struktura pro odpověď jednotlivých vrstev pevně definovaná v hlavním HTML dokumentu. Jinak řečeno, každá vrstva, ať už je přidána do mapové kompozice, nebo není, má v HTML dokumentu definovaný vzhled pro odpovědi. Může se tak stát, že uživatel nemá právo na zobrazení některé z vrstev, ale dokument bude strukturu odpovědi obsahovat. Celý tento proces tedy evidentně vyžaduje zásah na hned několika místech v kódu.

4.3 Uživatelské role a práva

Opomeneme-li zdlouhavý a komplikovaný proces rozšíření (nebo redukce) mapové kompozice, přichází v úvahu ještě jedna zásadní otázka - jak se v systému Floreon+ 2.0 řeší uživatelské role a práva? Identifikace uživatele v rámci systému, registrace uživatele a přihlašování již popisuje kapitola „WCF“, která vysvětluje technologické řešení autorizace a autentizace uživatele. Tato část textu přibližuje, jak funguje přizpůsobení uživatelského prostředí, pro již identifikovaného uživatele. Chování uživatelského rozhraní je vysvětleno na třech základních rolích v systému:

- Anonymní uživatel - uživatel bez vlastního profilu a identifikace v systému
- Registrovaný uživatel - uživatel s vlastním profilem a identitou v systému
- Administrátor - registrovaný uživatel s přístupem k rozšířeným možnostem systému

V otázce autorizace uživatele je největší újmou v systému Floreon+ 2.0 fakt, že webový klient je implementovaný v „client-side“ technologiích a tudíž většina operací se odehrává na straně uživatele. Z tohoto řešení také plyne, že uživatel má u sebe k dispozici celý zdrojový kód webového rozhraní a v případě uživatele technologicky zdatného, může hrozit nebezpečí neautorizovaného použití systému. Faktem je, že uživatelské rozhraní systému Floreon+ 2.0 je „napevno“ postaveno svou strukturou v HTML dokumentu. To znamená, že každý uživatel, který webové rozhraní navštíví, má strukturu HTML dokumentu totožnou a taktéž se za tohoto uživatele dotáží všechny mapové vrstvy na zdroj dat. To, v čem se posléze liší uživatelské rozhraní pro různé role uživatelů je implementovaná funkcionalita skrývání nepovolených elementů a funkcí.

Budeme-li konkrétně mluvit například o anonymním uživateli, bude se uživatelské rozhraní (resp. celý systém) chovat následovně. V momentě načítání webové aplikace v prohlížeči je uživatel rozpoznáván a hledá se validní *token* v *cookies* uživatele. U anonymního uživatele je *token* nenalezen a tudíž se uživateli přiřadí *token* označující nepřihlášeného uživatele. Takovýto *token* se uživateli do *cookies* prohlížeče neukládá, v případě přihlášeného uživatele se však po přihlášení *token* do *cookies* uloží. Klientská aplikace (webové rozhraní) zavolá službu WCF k určení práv uživatele a po přijetí odpovědi si informaci o právech uloží do své paměti (v případě webové aplikace je to operační paměť využívaná prohlížečem). V dalším běhu přípravy webového rozhraní se tyto práva kontrolují a pro funkcionality, které mají být právy omezené, se vykonává úkon spojený s jejich znepřístupněním danému anonymnímu uživateli. V případě mapových vrstev se vrstvy uživateli skryjí, nezobrazují se a například v případě analýz a predikcí se skryje tlačítko, kterým lze vyvolat jejich vytváření. Lze si snadno domyslet, že pro roli registrovaného uživatele se nebudou skrývat elementy, které se skrývají pro anonymního uživatele a stejně tak, že administrátorská role má přístup např. k mapovým vrstvám, které nejsou určeny pro veřejnost. Systém Floreon+ implementuje i mapové vrstvy vztahující se ke konkrétním událostem v Moravskoslezském kraji (jako např. MČR v ledním hokeji, nebo dny NATO), které byly ur-

čeny výhradně bezpečnostním složkám České republiky a tudíž existují v systému i specifické uživatelské role sloužící právě těmto státním orgánům.

4.4 Omezení systému

Shrnutím webového rozhraní systému Floreon+ 2.0 dojdeme k jednoduchému závěru - změna mapové kompozice ve smyslu přidání, úpravy nebo odebrání mapové vrstvy je poměrně zdlouhavý a komplikovaný proces. K přidání nové mapové vrstvy je potřeba zásah programátora na několika místech do zdrojového kódu webového rozhraní a řešení je velmi problematické na úpravy - pokud se změní například pojmenování vrstvy v GeoServeru, dojde k porušení funkčnosti dané vrstvy na webovém rozhraní systému. Stejně tak, pokud by se měla mapová kompozice rozšířit o novou kategorii vrstev, je nutné manuální vytvoření kategorie ve zdrojovém kódu a posléze přidávání vrstev do této kategorie. Ovládací mapový prvek „LayerSwitcher“ funguje s mapovými vrstvami, které jsou již součástí mapové kompozice - tudíž není přímo závislý na konkrétní definici mapové vrstvy. Dalším problémem řešení webového rozhraní Floreon+ 2.0 je funkcionalita související s uživatelskými právy. Ačkoliv klíčové operace a funkcionality jsou pokaždé verifikovány uživatelským *tokenem*, považuji za bezpečnostní riziko načítat uživateli všechny struktury a posléze je podle práv pouze skrývat nebo odstraňovat. Další skutečností je fakt, že i uživatel, který nemá oprávnění na například jisté mapové vrstvy, má tyto vrstvy ve svém řešení implementované a přidané do mapové kompozice. Taktéž probíhá mnohdy volání tohoto uživatele na GeoServer nebo WCF službu, které nemá opodstatnění, neboť i kdyby služba data vrátila, uživatel by si je nemohl zobrazit. Taktéž i z pohledu verze frameworku OpenLayers 2.0 je nutná aktualizace a přechod na novější, udržovanou verzi OpenLayers 3.0, neboť by se webové rozhraní a především tedy mapová kompozice mohla v brzké době potýkat s nekompatibilitou webových prohlížečů a použité technologie.

5 Floreon+ 3.0

Ze sekce 4 a konkrétně podkapitoly 4.4 lze snadno pochopit hlavní motivaci a zadání této práce. Následující kapitola se věnuje úpravám, které se na webovém rozhraní systému Floreon+ prováděly. Ačkoliv zadání práce samotné je zaměřeno na dynamické generování mapové kompozice, bylo nutné celý koncept skládání webového rozhraní přestavět na dynamický přístup. Důvod je poměrně jednoduchý a logický - je zbytečné mít v HTML dokumentu webového rozhraní struktury pro zobrazení výsledku vrstev, které skupina uživatelů nikdy neuvidí. Stejně tak jako je zbytečné, a pouze to zpomaluje chod webového klienta, vytvářet v mapové kompozici vrstvy, na které posléze uživatel nebude mít právo je vizualizovat. Výsledné webové rozhraní má tedy být schopno sestavovat každému uživateli mapovou kompozici podle jeho oprávnění, stejně tak jako strukturu dokumentu a taktéž přizpůsobit ovládací prvek nově vygenerované mapové kompozice podle práv uživatele.

Webové rozhraní systému Floreon+ se skládá z hlavního HTML souboru, několika CSS souborů a taktéž z mnoha JavaScriptových souborů. Důvodem rozdělení JavaScriptového kódu do více souborů je snaha oddělit jednotlivé funkcionality systému. Např. soubor *map.js* je soubor, který nás bude zajímat nejvíce, zatímco soubor *ui.js* nás téměř zajímat nebude, neboť tento soubor se skripty ovlivňuje hlavně chování webového rozhraní bez (nebo s minimální) návazností na mapový podklad. Naopak soubor *map.js* je téměř výhradně určený k operacím nad mapovou kompozicí. Ve skriptech tohoto souboru se vytváří hlavní element *map*, který inicializuje objekt mapy do zadaného HTML elementu.

Vzhledem k tomu, že je tedy prakticky potřeba implementovat od začátku jádro mapové kompozice webového rozhraní systému, byl vytvořen seznam priorit úkonů, které bylo potřeba postupně implementovat po změně verze OpenLayers:

1. Vytvořit prázdný mapový element
2. Přidat do mapové kompozice některé mapové vrstvy staticky
3. Implementovat ovládací prvek vrstev - „LayerSwitcher“
4. Vytvořit pomocné objekty mapových vrstev
5. Vytvořit proces mapování *capabilities* na pomocné mapové objekty
6. Vytvořit proces přetvoření pomocných mapových objektů na objekty mapových vrstev
7. Proces události kliknutí na vybrané vrstvy
8. Interakce rozhraní s mapovou kompozicí

5.1 OpenLayers 3

V první části úprav tedy šlo o nahrazení JavaScriptových souborů OpenLayers verze 2 soubory OpenLayers 3 a vytvoření nového souboru *map.js*. Po těchto krocích přestalo webové rozhraní kompletně fungovat. Pokud bychom nechali původní soubor *map.js*, webové rozhraní by spadlo, respektive vůbec nenaběhlo, neboť spousta proměnných a metod z verze OpenLayers 2 neexistuje a nebo existuje, ale s jinou anotací. Bylo tudíž mnohem jednodušší, vytvořit JavaScriptový kód, obsluhující framework OpenLayers znovu.

Verze OpenLayers 3 byla vyvinuta nezávisle na kompatibilitě s dřívější verzi, což způsobí uživatelům při přechodu mnoho nepříjemností, naopak nová verze je v mnoha ohledech lépe postavená a využívá nejmodernějších technologií.

Podívejme se tedy na nejvýraznější změny OpenLayers 3 oproti OpenLayers 2.

5.1.1 Vektorová architektura

Přepracovaná vektorová architektura silně posilnila vektorové části mapy, kdy tedy nově dochází k překreslení všech vektorových elementů v momentě kdy se překresluje mapa. Typicky tak dochází k překreslování vektorových elementů 30krát až 60krát do sekundy, když uživatel s mapou aktivně zachází. Výsledkem této úpravy je vysoká kvalita vykreslování vektorových dat, které se tak vykreslují podle rozlišení a např. rotace mapy. Vektorové elementy jsou taktéž překreslovány během animací a interakcí v mapové komponentě.

Dalším místem kde lze toto vylepšení pozorovat je „kreslení“ geometrických vektorových ploch, typicky využívané v systému Floreon+ pro zadávání analýz a predikcí. Nová vektorová architektura zajišťuje lepší výkon kreslení, spolu s lepší kvalitou (vektorový objekt je lépe přizpůsobený rozlišení obrazovky). Spolu s „kreslením“ vektorových ploch stojí za zmínku API, které umožňuje „kreslení“ bez inicializace nové vektorové vrstvy. V OpenLayers 2 bylo vždy třeba vytvářet pomocnou vektorovou vrstvu, do které se plocha zájmu vykreslila a po využití, se opět vrstva z mapové kompozice mazala.

Celková vektorová architektura byla přetvořena podle WebGL[15] technologie. Ačkoliv OpenLayers 3 technologii WebGL ještě plně nevyužívá, novou vektorovou architekturou, si však otevírá dveře do dalších verzí.

5.1.2 High-DPI podpora

Tato úprava taktéž souvisí s kvalitou vykreslování mapové kompozice. Díky této novince, umožňuje OpenLayers vykreslování WMS a WFS vrstev ve vyšší kvalitě (WFS také díky vektorové architektuře). Z klienta dochází k posílání požadavků o větší a kvalitnější obrázky (pokud je GeoServer schopen je poskytnout) a jejich vykreslování do většího *canvas* objektu (*canvas* je element používaný k vykreslování grafiky ve webové stránce). Tato úprava souvisí s moderní zobrazovací technikou, kdy rozlišení klientů dnes umožňuje využívat ta nejvyšší rozlišení aplikací.

[15] https://www.tutorialspoint.com/webgl/webgl_introduction.htm

Pokud je vlastnost *window.devicePixelRatio* (vlastnost zobrazovacího zařízení získaná pomocí JavaScriptu) větší než 1, začíná OpenLayers 3 využívat tohoto rozšíření.

5.1.3 Parsování formátů

Tato úprava rozšiřuje škálu formátů, které lze pomocí OpenLayers vizualizovat. Jedná se o úpravy „parserů“ a samotných formátů podle nejnovějších specifikací daného standardu. Typickými příklady pak mohou být formáty KML[16], GeoJSON, TopoJSON[17] nebo GPX[18].

5.1.4 Shrnutí

Ve výsledku tedy můžeme říci, že nová verze je zaměřena na výkon frameworku a také kvalitu, kterou umožňují nejnovější technologie. Na závěr o nové verzi OpenLayers také můžeme zmínit, že má mnohem lépe zpracovanou API dokumentaci [1] spolu se spoustou příkladů, které uživateli velmi usnadní práci s frameworkem, nebo s přepisem stávající aplikace na novou verzi OpenLayers.

5.2 LayerSwitcher

Vrátíme se tedy k implementaci systému Floreon+ 3.0. Jak již bylo zmíněno na začátku této kapitoly, byl vytvořen seznam úkolů, potřebných pro přechod ze systému Floreon+ 2.0 na Floreon+ 3.0. a prvním úkonem tedy bylo vytvoření prázdného elementu mapy a posléze přidání statické mapové vrstvy.

5.2.1 Inicializace mapové komponenty

Prvním úkolem tedy bylo vytvořit prázdný mapový element. V dokumentu *index.html* je definovaná oblast označená jako *map-container*, kde tato oblast je umístěna přes celý rozměr obrazovky a má sloužit jako prostor pro mapovou kompozici. Všechny ostatní HTML elementy (pravý vysouvací panel, dolní lišta, časová osa, apod.) jsou elementy umístěny „nad“ touto oblastí určené k mapové kompozici. Pomocí jQuery definujeme událost načtení HTML dokumentu, tedy moment, kdy celá statická struktura je načtená a zobrazená. Je důležité upozornit na slovo statická, znamená totiž, že se načetly zatím části webového rozhraní, které jsou explicitně napsány v HTML. Pokud by se nespustily žádné skripty, webové rozhraní by bylo nekompletní, téměř žádné funkcionality by nebyly k dispozici a navíc celé webové rozhraní by bylo nepřeložené, tzn. že celý dokument by obsahoval pouze „tagy“, které jsou běžně při načítání webu nahrazovány výrazy podle vybrané jazykové lokalizace. Jakmile je dokument staticky načtený, dojde k události, pomocí které můžeme implementovat další chování systému. Webové rozhraní

[16] https://cs.wikipedia.org/wiki/Keyhole_Markup_Language
[17] https://github.com/topojson/topojson
[18] https://en.wikipedia.org/wiki/GPS_Exchange_Format

systému Floreon+ se skládá hned z několika JavaScriptových souborů, kde téměř každý definuje událost načtení webového rozhraní a na základě této události spouští procesy, pro které je definován. Například soubor *ui.js* po statickém načtení HTML dokumentu spouští procesy pro úpravu webového rozhraní („personalizace“, překlad, atd.). Mapová kompozice se však nejvíce váže k souboru *map.js* a zde se událost načtení webového rozhraní implementuje také. Veškerou funkcionalitu spjatou s načtením webového rozhraní zajišťuje funkce *initMapElements*, která v aktuální fázi celkového vývoje již obsahuje více dalších volání, nicméně pořád to nejprioritnější je volání funkce *mapInit*.

V této funkci totiž definujeme objekt *ol.Map*, který je nejdůležitějším objektem celého systému. Reprezentuje kompletní mapovou kompozici, spolu se všemi událostmi, interakcemi, vrstvami apod. Jelikož se vždy hodí mít možnost přistupovat k objektu mapové kompozice odkudkoliv z kódu, byla vytvořena globální proměnná *map*, které posléze byla přiřazena instance vytvořené mapové kompozice. Při vytváření objektu mapy se muselo definovat několik základních parametrů, jako jsou umístění do HTML elementu, základní nastavení zobrazení (maximální a minimální povolený zoom na mapě, výchozí interakce, výchozí ovladače mapové kompozice a také výchozí mapové podklady). Pochopitelně všechny tyto atributy lze v průběhu modifikovat, přidávat a odebírat. Pro potřeby počáteční inicializace mapové komponenty byla nastavena jako výchozí mapový podklad vrstva Open Street Map, která byla definována jako WMS. Po načtení webového rozhraní se již zobrazí mapová komponenta s OSM vrstvou, avšak žádné funkcionality systému spjaté s mapovou komponentou fungovat nebudou. Hned první věc, které jsme si mohli všimnout, bylo, že OpenLayers neposkytuje komponentu „LayerSwitcher“, kterou ve verzi 2 implementoval. Jinak řečeno, verze 3 neobsahuje komponentu na ovládání mapových vrstev v kompozici, tak jako obsahovala předchozí verze. Ačkoliv elementární prvek interakce s mapovými vrstvami ve verzi OpenLayers 3 existuje, neodpovídá svou koncepcí způsobu ovládání mapových vrstev v systému Floreon+.

5.2.2 Interaktivní ovládací prvek

Jelikož hlavním zaměřením této práce bylo vytvoření dynamicky generované mapové kompozice, bylo potřeba implementovat ovládací prvek systému, ve kterém se vrstvy přidané v mapové kompozici objeví. Bez takového prvku by nemohlo být kontrolováno, které mapové vrstvy již kompozice „zná“ a umí s nimi zacházet a které se nepodařilo správně přidat, nebo se s nimi zachází špatně. Název, který bude nadále používán pro takový funkční prvek je „LayerSwitcher“ nebo ovládací prvek mapové kompozice. Jak už je z názvu zřejmé, jedná se o prvek, který pozná, jaké mapové vrstvy jsou k dispozici a poskytne možnost manipulace s nimi. Implementace ovládacího prvku byla realizována v souboru *layerswitcher.js*. Vzhledem k faktu, že ve verzi Floreon+ 2.0 byl „LayerSwitcher“ implementován, bylo nutné aby se styl a funkčnost ovladače držely stejného konceptu. Muselo se tedy jednat o výčet všech vrstev dostupných v mapové kompozici, které zároveň jsou rozděleny do kategorií dle obsahu. Jak jednotlivé kategorie, tak i samotné vrstvy, musí být automaticky generované podle poskytovaných služeb GeoServerem.

OpenLayers umožňuje vytvářet objekty typu *group*, do kterých lze přidávat jednotlivé mapové vrstvy. Při další implementaci bylo nutné využít tohoto dělení mapových komponent.

5.2.2.1 Seznam vrstev Za předpokladu, že tedy skládání mapové kompozice bude probíhat vytvořením všech skupin vrstev a každá skupina posléze bude naplněná vrstvami s obsahem relevantním k tématu, bylo vhodné „LayerSwitcher“ implementovat stejným způsobem. Ideálním HTML prvkem poskytující danou funkčnost je odrážkový seznam. Odrážkový seznam umožňuje definovat více-úrovňovou strukturu. Byla tedy implementována funkce, která v parametru přebírá množinu objektů reprezentujících vrstvu nebo skupinu vrstev. Tato funkce projde předanou kolekci a zjistí, zda se jedná o skupinu nebo o samotnou vrstvu. Tato kontrola je důležitá hlavně z toho důvodu, že v OpenLayers 3 se jak skupina, tak samotná vrstva, reprezentují stejným objektem, liší se pouze stavem (vlastnostmi) objektu. Funkce zároveň vygeneruje oblast označenou příslušným identifikátorem, do které dynamicky vytvoří DOM element *ul*, neboli odrážkový seznam. Pro každou nalezenou skupinu vrstev funkce „LayerSwitcheru“ vytvoří vlastní *li* element, který jednoduše reprezentuje odrážku. Takto funkce v prvním běhu vytvoří seznam všech skupin vrstev, které mapová kompozice má načtené. Ovšem dále bylo potřeba, aby skupiny vrstev byly naplněny samotnými vrstvami. K tomuto účelu slouží identická funkce, kde vstupním parametrem (kolekce vrstev) je jednotlivá skupina vrstev. Toto rekurzivní volání bylo přidáno do funkce, kdy funkce v prvním běhu cyklu nalezne skupinu vrstev, vytvoří pro ně odrážku a rekurzivně zavolá sama sebe. Výsledkem tedy je odrážkový seznam, kde v první úrovni máme odrážky obsahující seznamy vrstev a každá takováto odrážka, pokud obsahuje vrstvy, má další „pod“ seznam, který má stejné členění. Čistě teoreticky, pokud by seznam vrstev obsahoval například pět samostatných vrstev a jednu další skupinu vrstev, takto implementovaný algoritmus bude schopný výslednou strukturu korektně zachytit a vizualizovat.

5.2.2.2 Funkcionalita seznamu Pokud jsme měli korektně zachycenou hierarchii skupin vrstev a samotných vrstev, bylo potřeba implementovat do této komponenty funkčnost. Aby byl zachován koncept z Floreon+ 2.0, musel být implementován rozbalovací seznam, kde skupiny vrstev slouží jako kategorie, které při kliknutí zobrazí samotné vrstvy, jež skupina obsahuje. U samostatných vrstev bylo pak potřeba implementovat HTML element *checkbox*, který zcela intuitivně má znázorňovat, zda je příslušná vrstva zobrazená nebo skrytá. Vytvořenou strukturu vrstev bylo pochopitelně potřeba správně označit třídami a identifikátory, které velmi usnadní další úpravy jednotlivých položek seznamu. Každé skupině vrstev byla přiřazena třída *group* a každé samostatné vrstvě třída *layer*. Přiřazení tříd bylo ovšem taktéž nezbytné pro design celého ovládacího prvku pomocí CSS. Pro vytvoření události „kliknutí“ na jednotlivou položku seznamu vrstev byla opět rozšířena funkce, která celou strukturu generuje. Jelikož již byly rozlišeny skupiny vrstev a jednotlivé vrstvy, snadno se dalo rozvětvit chování algoritmu a mohlo tak být přiřazeno události kliknutí na skupinu vrstev jiné chování, než při kliknutí na samostatnou vrstvu.

Byla vytvořena tedy pro každou z těchto dvou případů událost kliknutí a navázaná funkcionalita. Pokud byla vyvolána událost kliknutí na skupinu vrstev, spustí se algoritmus, který v první fázi zjistí, zda daná skupina vrstev obsahuje ve svých potomcích (DOM hierarchická struktura) element *ul*. Pokud takový element existuje, jsou dvě možnosti - buď je skrytý a tudíž jej funkce zviditelní a všechny ostatní *ul* elementy skryje, nebo již je zobrazený a v takovém případě jej funkce skryje. Pokud takový element neexistuje, nestane se nic, a taky nám vzniká prostor na optimalizaci, neboť zobrazená prázdná skupina vrstev nemá žádný relevantní význam a může být z ovládacího prvku odstraněna.

Druhou větví algoritmu muselo být definováno chování pro kliknutí na element reprezentující samostatnou mapovou vrstvu. Při takové události zcela intuitivně očekáváme, že vybraná vrstva se zobrazí/skryje v mapové kompozici. Jelikož každý element ovládacího prvku byl vygenerován podle příslušné vrstvy (nebo skupiny vrstev), mohli jsme velmi jednoduše definovat, které vrstvy se kliknutí týká. Jelikož jsme měli přístup k samotnému objektu mapové vrstvy, který implementuje funkci *setVisible*, která boolovským parametrem vrstvu buď zobrazí nebo skryje, nebyla zde nejmenší obtíž tuto funkcionalitu navázat na událost kliknutí. Kdybychom se ovšem zamysleli, zjistili bychom, že je pro nás výhodnější definovat si pomocnou funkci na úrovni „LayerSwitcheru“, která umožní kontrolovat zobrazování a skrývání mapových vrstev. Jedním z typických důvodů může být například skupina vrstev „basemaps“, neboli podkladové mapy. Jelikož podkladová mapa, jak už její název vypovídá, má být „nejnižší“ vrstvou v mapové kompozici, je zbytečné, aby byly zapnuté další podkladové vrstvy - navzájem by se překrývaly. Díky této funkci tak mohl být efektivně implementován algoritmus, který při zobrazení podkladové vrstvy skrývá všechny ostatní.

V této fázi implementace byla přidána pouze jedna vrstva a navíc staticky. Pro ověření korektního chování bylo možné vytvořit několik skupin vrstev, do každé z nich přiřadit například duplicity již přidané OpenStreet Map vrstvy a skupiny vrstev přidat do mapové kompozice pomocí funkce *addLayer*, kterou implementuje objekt *map*.

5.2.2.3 Vzhled ovládacího prvku Ačkoliv tato část textu je pojmenována „Vzhled ovládacího prvku“, nebude tato část textu o CSS designu, respektive jen okrajově. Máme vytvořenou „oblast“, která se po načtení webového rozhraní plní strukturou mapové kompozice a také nám poskytuje možnost skrze své funkce manipulovat se skladbou mapové kompozice. Již v předchozí části textu bylo zmíněno, že bylo potřeba implementovat k jednotlivým položkám DOM elementy, které ponesou informaci, zda je vrstva zobrazená nebo skrytá. Pro vrstvy, které mohou být zobrazeny paralelně, je takovým typickým ukazatelem element *checkbox*, avšak například pro již zmíněné podkladové vrstvy, by *checkbox* postrádal smysl.

Proto do algoritmu, který skládá strukturu vrstev, bylo přidáno ještě jedno malé vylepšení. Trochu předběhněme a podívejme se na informace, které bylo potřebovat vědět o mapových vrstvách, aby mohly být přidány do mapové kompozice. Abychom byli schopni se korektně doptávat GeoServeru na jednotlivé vrstvy, museli jsme znát jejich název v GeoServeru, dále

jsme museli znát „namespace“ vrstev, což je prakticky pojem, označující množinu vrstev se společnými rysy a zájmovou oblastí. Pochopitelně jsme museli znát spoustu dalších atributů, avšak v dané fázi implementace nám stačily tyto dva.

Jmenný prostor vrstev byl tedy považován za název skupiny, do které vrstvy patří a jejich název na GeoServeru byl použit jako název vrstev v již implementovaném ovládacím prvku. Ve chvíli, kdy jsme měli tyto informace o jednotlivých vrstvách, mohly být implementovány podmínky, podle kterých se určí, kdy se k dané položce v seznamu vrstev přiřadí element *checkbox* a kdy element *radio button*. Pokud vrstva patří do jmenného prostoru (a tudíž posléze i do skupiny) s názvem „basemaps“, vygeneruje jí funkce jako pomocný element *radio button*. Další úpravou byla událost kliknutí, která byla navázána místo celého elementu na nově vygenerovaný *checkbox* u položek vrstev. Taktéž ovšem musela být rozšířena funkce v tom smyslu, že pokud při načtení mapové kompozice byla některá vrstva ve výchozím nastavení zobrazená, aby byl stejně tak i označený příslušný *checkbox* vrstvy. Vzhledem k tomu, že objekt mapy implementuje událost při změně ve své kompozici, byla navázána na tuto událost funkce přegenerování ovládacího prvku, čímž bylo zaručeno, že kdykoliv se do mapové kompozice přidá nová vrstva, nebo se z ní vrstva odebere, bude ovládací prvek aktuální a bude obsahovat jen skutečné vrstvy existující v kompozici. Předposlední drobnou úpravou byla lokalizace. Jelikož dopředu není znám počet skupin vrstev, vrstev samotných, ani dokonce jejich názvy, bylo nutné implementovat obecný algoritmus pro překládání názvů těchto prvků. Každý název skupiny vrstev i samotných vrstev byl umístěn ještě do DOM elementu *span*, kterému byla přiřazena třída *lang*. Tato třída slouží k označení textového elementu, který se má při načítání webového rozhraní nahradit příslušným výrazem z vybraného jazyka. O lokalizaci webového rozhraní budeme ještě mluvit později, avšak tuto část bylo nutné zmínit již zde.

Po těchto krocích byl implementovaný plně funkční ovládací prvek, který byl navíc připravený pro lokalizaci, dle vybraného jazyka. Posledním úkolem bylo tedy celý tento ovládací prvek umístit do webového rozhraní. Od systému Floreon+ 2.0 existuje v rozhraní tlačítko reprezentující „LayerSwitcher“, a protože od začátku celá struktura je skládána do DOM oblasti (element *div*), velmi jednoduše byla implementována oblast obsahující celý nový „LayerSwitcher“ jako DOM potomka oblasti, která ve verzi 2.0 implementovala samotný „LayerSwitcher“ objekt.

5.3 Zdroj dat capabilities

Byla tedy vytvořena mapová kompozice a vytvořený ovládací prvek „LayerSwitcher“, který umožňoval v rámci mezí mapovou kompozici ovládat. Do mapové kompozice bylo pro otestování funkčnosti ovládacího prvku přidáno několik skupin vrstev, které obsahovaly jednotlivé objekty vrstev. Otázka, která se tedy naskytla, zněla - jakým způsobem lze naplnit mapovou kompozicí mapovými vrstvami, na které má uživatel práva? Odkud vzít data potřebná k sestavení korektního dotazu na služby GeoServeru tak, aby GeoServer vrátil mapovou vrstvu?

Odpovědí byl opět GeoServer, respektive odpověď na tyto otázky jsou již zmíněny v podkapitole 3.2.3. Tato část textu rozebírá podrobně strukturu XML dokumentu *capabilities* s ohledem

na následné využití v algoritmech sloužících ke generování mapové kompozice. V první řadě bylo potřeba si vyžádat od instance GeoServeru soubor s *capabilities*. To bylo provedeno poměrně jednoduše, například ve webovém prohlížeči nebo pomocí nástroje SoapUI (nástroj na provolávání a testování webových služeb)[19].

Pro tyto účely byly využity *capabilities* z jednoho z vývojových GeoServerů. URL má následující tvar - *http://<urlGeoServer>/GeoServer/ows?* je umístění GeoServeru a následují parametry požadavku typu *GET - service=wms,version=1.3.0,request=GetCapabilities*, a nakonec identifikační klíč *authkey=<token>*. Jak si můžeme všimnout, v URL se používají všechny parametry, jako *service,version,request* a pro tento účel i *authKey*. Taktéž snadno z URL vyčteme, že se jedná o *capabilities* WMS standardu, pokud bychom chtěli *capabilities* WFS standardu, stačilo by parametr *service* zaměnit za WFS.

5.3.1 Struktura XML souboru

Výsledkem provolání správně sestaveného URL je tedy XML soubor obsahující *capabilities*. Tento soubor obsahuje téměř všechny relevantní informace spjaté s mapovými vrstvami ve formátu WMS (strukturu XML dokumentu lze vidět na obrázku 7). Pro představu vzniklo znázornění alespoň části základní struktury dokumentu, kterou můžeme vidět níže.

- Service
- Capability
  - Request
    - GetCapabilities
    - GetMap
    - GetFeatureInfo
  - Exception
  - Layer

V kořenu dokumentu jsou dvě hlavní části, kterými jsou „Service“ a „Capability“. „Service“ obsahuje pouze informace o poskytované službě - tedy informace jako název, abstrakt, klíčová slova, název provozující instituce a kontaktní údaje. Část Capability“ se dále dělí na tři části, pro mou implementaci ta nejdůležitější je část „Layer“. XML element „Request“ obsahuje výčet požadavků, kterými lze volat služby GeoServeru. Element „Exception“ obsahuje pouze formáty, ve kterých lze očekávat výjimky. Nakonec element „Layer“, ten je nejrozsáhlejší částí celého XML dokumentu. Největší část obsahu zabírá kolekce dalších „Layer“ elementů, kde ale každý jeden prvek kolekce reprezentuje jednu mapovou vrstvu formátu WMS, kterou je GeoServer schopný poskytnout. Vzhledem k tomu faktu, že požadavek o *capabilities* byl poslán již s autorizačním

[19] https://www.soapui.org/

klíčem (*tokenem*), odpovídá i počet vrstev v této kolekci množině vrstev, které může uživatel s daným identifikátorem zobrazit.

5.3.2 Element vrstvy

Nejdůležitější pro implementaci je tedy kolekce elementů „Layer“, která je potomkem elementu „Layer“ (pojmenování může být matoucí), kde každý prvek kolekce reprezentuje jednu WMS vrstvu, kterou chceme přidat do mapové kompozice.

Obrázek 7: XML struktura vrstvy kamer

Na obrázku 7 lze vidět XML strukturu jedné z mapových vrstev, konkrétně vrstvy dopravních kamer. Vyznačené elementy označují nejdůležitější informace o vrstvě. Hned v kořenovém elementu celé vrstvy si lze všimnout důležitého atributu „queryable“, který označuje, zda daná

vrstva obsahuje informace po kliknutí na element v ní. Jinak řečeno, je-li hodnota atributu „queryable“ u vrstvy rovna 1, pak ve chvíli, kdy je vrstva zobrazena v mapové kompozici a je na ni vytvořená událost kliknutí, existují data v GeoServeru, které lze poskytnout.

Element „Name“ nese nejdůležitější informaci, a to název vrstvy v GeoServeru. Navíc se název skládá z jmenného prostoru a názvu vrstvy oddělené dvojtečkou, kde jmenný prostor prakticky reprezentuje skupinu vrstev. Na obrázku lze vidět, že element „name“ obsahuje vrstvu „rodos:camera“, z čehož lze snadno získat informaci, že tato vrstva patří do skupiny vrstev „rodos“, čemuž odpovídají data relevantní k dopravě. Elementy jako „title“ nebo „keywordlist“ k vytvoření vrstvy v mapové kompozici nepotřebujeme, ale stejně je vhodné si tato data k vrstvám ukládat. Kolekce elementů „BoundingBox“ označuje v závislosti na souřadnicovém systému jakou mapovou oblast daná vrstva pokrývá. Je logické, že pokud máme vrstvu s dopravními kamerami na území města Ostravy, není potřeba, aby vrstva obsahovala mapové informace (které by stejně byly prázdné) pro například zbytek celé České republiky. Dalšími důležitými elementy jsou „Attribution“ a „Style“. „Attribution“ uchovává informace o autorovi dat, nebo poskytovateli určitých informací, obsáhlých v dané vrstvě. Na základě autorských zákonů je nutné tyto informace zobrazovat. Poslední element, který si zmíníme je element „Styles“, který nese informace o stylech vrstvy. Každé vrstvě lze nadefinovat hned několik stylů, kterými lze posléze modifikovat vzhled poskytnutých dat.

5.4 Dynamická mapová kompozice

V této fázi implementace jsme tedy věděli, kde leží data o přístupných mapových vrstvách pro jednotlivé uživatele systému, věděli jsme také, jak se k těmto datům dostat a měli jsme přesnou představu o tom, jakou strukturu data mají. Taktéž zde již byla připravená prázdná mapová kompozice s implementovaným dynamickým ovládacím prvkem, která v tu chvíli obsahovala pouze staticky přidané testovací mapové vrstvy. K sestavení mapové kompozice z dostupných dat získaných pomocí *capabilities* GeoServeru bylo potřeba sestavit proces, který si umí automaticky data načíst a najít si v nich potřebné informace. V první řadě se muselo v rámci tohoto procesu zjistit, jaké všechny skupiny vrstev se vyskytují v dostupných datech a podle této informace sestavit skupiny vrstev v mapové kompozici. Posléze se musela implementovat funkčnost, která v datech vyhledá jednotlivé vrstvy a vhodným způsobem je transformuje na objekty mapových vrstev, které se přidají do příslušných skupin.

V procesu musela být zohledněna skutečnost, že některé vrstvy, které jsou uživateli dostupné, není potřeba zobrazovat v ovládacím prvku. Může se jednat o pomocné mapové vrstvy, které obsahují informace relevantní k některému z procesů, avšak celkové jejich zobrazení postrádá smysl. Typickým příkladem může být vrstva adres. Jelikož systém Floreon+ umožňuje vyhledávání adres a tzv. „routing“ (vyhledávání tras), existuje mapová vrstva obsahující informace s adresami. Zobrazení celé vrstvy by znamenalo zobrazení všech existujících adres jako označených ve výběru. Takováto vrstva se tedy používá v rámci některého z procesů, kde například, zadá-li uživatel adresu do vyhledávání, výsledkem bude zvýraznění místa na mapě odpovídající zadané

adrese. Z tohoto důvodu takovýto typ vrstev není vhodné zobrazovat v „LayerSwitcheru“, neboť při zapnutí vrstva nenese žádná relevantní data, naopak může uživatele obtěžovat a mást. Jakmile byla sestavena hierarchie skupin vrstev a vrstev samotných, musela být implementována funkčnost dotazování na data jednotlivých vrstev, jež dotazování umožňují. V souvislosti s tímto úkolem bylo potřeba implementovat dynamické generování DOM struktur pro zobrazování získaných dat v pravém panelu webového rozhraní.

Nakonec bylo potřeba přepsat funkce spjaté s ovládáním mapové kompozice, jako například časová osa, legenda, měření v mapové kompozici a další.

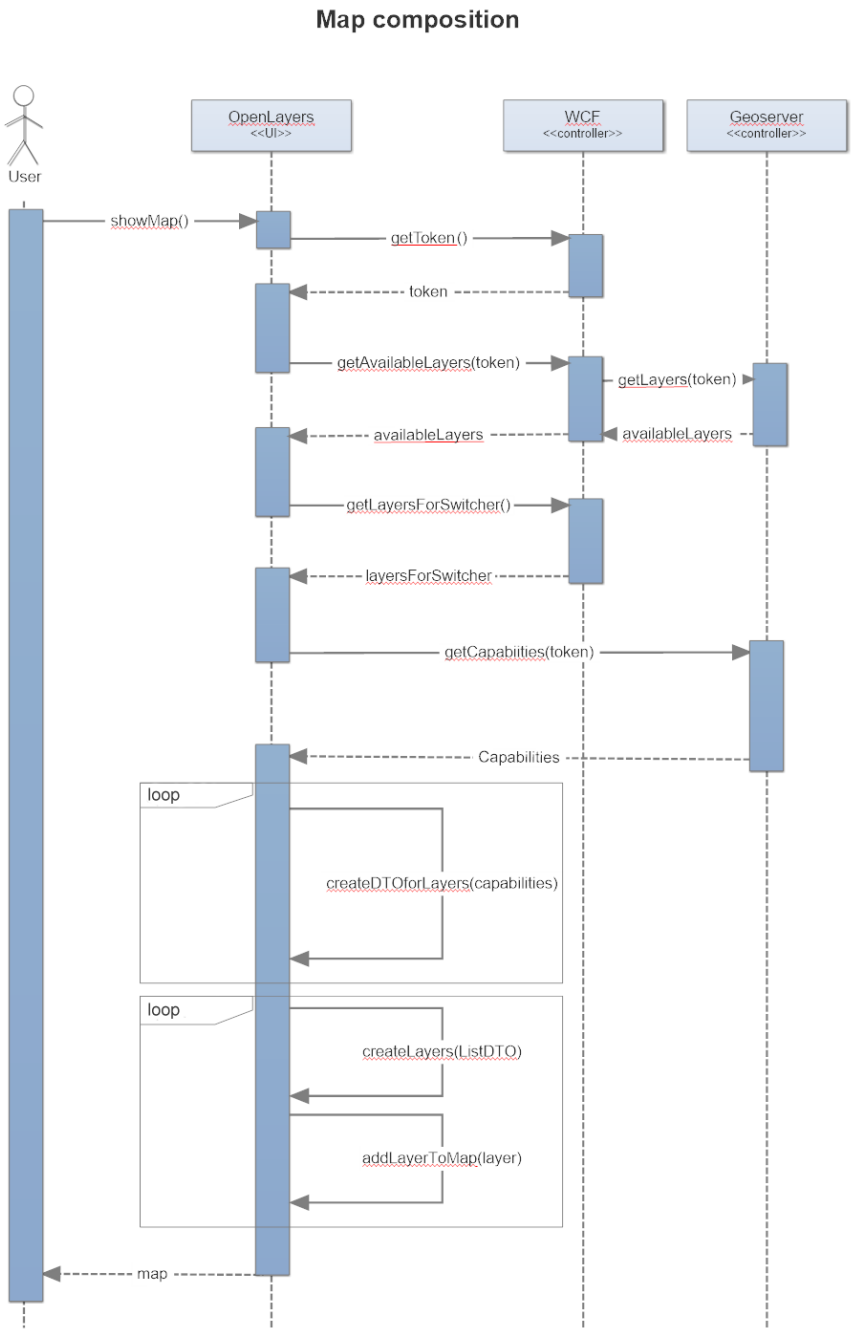
5.4.1 Návrh řešení

Berme tedy v potaz, že existují tři prvky systému, které se musí podílet na procesu generování mapové kompozice. Jsou to OpenLayers framework, WCF a GeoServer. Jakmile uživatel navštíví webové rozhraní, odesílá webový klient dotaz na WCF, aby získal identifikační *token*. Pokud WCF služba daného uživatele nezná, vrací zpět klientovi *token* určený pro anonymní uživatele. Dalším krokem je doptání se WCF na dostupné mapové vrstvy pro uživatele s tímto *tokenem*. Pokud se ptáte, proč je dotaz směřován na službu WCF a ne na GeoServer *capabilities*, je to z toho důvodu, že v řešení bylo lepší, implementovat možnost omezení množiny mapových vrstev na úrovni WCF. Z mého pohledu je potřebné mít možnost ovlivnit zobrazovanou kompozici i jiným způsobem, než úpravou práv jednotlivých uživatelů. Na úrovni webového klienta je tedy touto implementací získána první množina názvů vrstev, které jsou uživateli k dispozici.

Výsledkem dalšího dotazu na WCF službu je množina vrstev, které mají být zobrazeny v ovládacím prvku „LayerSwitcher“. Tato informace se získává z GeoServeru, kde je implementována role „LayerSwitcher“ určující množinu vrstev, které se mají zobrazovat v ovládacím prvku „LayerSwitcher“. Reálně pak není tato role přiřazena žádnému uživateli systému a slouží pouze k definování množiny vrstev. Výsledkem dotazu je tedy druhá množina vrstev, která může být podmnožinou první množiny.

Poslední dotaz webového klienta směřuje na GeoServer, službu *GetCapabilities*. Odpovědí je kompletní XML soubor, obsahující (nejen) elementy mapových vrstev, na které má daný uživatel práva. Celý tento proces vystihuje sekvenční diagram na obrázku číslo 8.

Diagram zachycuje i další proces, který již pracuje se získanými daty na straně klientské aplikace. Vezmeme-li v úvahu, že v aktuálním stádium procesu existovaly dvě množiny dat označující vrstvy, které mají být přidány do mapové kompozice a načtený XML soubor se všemi relevantními údaji k jednotlivým vrstvám, bylo potřeba dále implementovat funkčnost, která získaná data transformuje na objekty vrstev frameworku OpenLayers. Prakticky proces vykonává činnost mapování XML elementů vrstev přímo na objekty vrstev OpenLayers, avšak součástí návrhu je i mezivrstva v podobě DTO. DTO jsou objekty obsahující pouze parametry a funkce, které dané parametry umí nastavit nebo vrátit při zavolání. V prvním cyklu mapování proběhne namapování elementů vrstev z XML souboru na tyto DTO objekty. V podstatě tato implementace první řadě vygeneruje kolekci takovýchto DTO entit, které obsahují všechna data



Obrázek 8: Sekvenční diagram mapové kompozice

potřebná k vytvoření mapových vrstev. V dalším cyklu procesu se tyto DTO entity přemapují na konkrétní mapové vrstvy a ty se posléze přidají do mapové kompozice. Návrh obsahuje tuto mezivrstvu z důvodu budoucích možných aktualizací frameworku OpenLayers a změny anotace mapových vrstev v tomto frameworku a také z možnosti aktualizace GeoServeru, kde se může potenciálně měnit struktura *capabilities* dokumentu. Taktéž navíc tato vrstva umožňuje vytvo-

ření mapových vrstev z jiného zdroje. Aktuálně systém Floreon+ využívá pro všechny mapové podklady GeoServer, nicméně v dalších verzích systému může přibýt další podobný nástroj a vrstva DTO v takovém případě bude prostor, kde se procesně vyřeší mapování dat z nového zdroje na DTO entity a proces mapování těchto entit na mapové vrstvy nebude ovlivněn.

Když je vytvořena kolekce všech objektů vrstev, provede se další cyklus procesu, který rozřadí kolekci do příslušných skupin vrstev, kde každá skupina odpovídá jednomu jmennému prostoru v GeoServeru. V rámci tohoto cyklu se přiřadí každé vrstvě označení *showInLayerSwitcher*, které se posléze implementuje do ovládacího prvku. Jakmile proces dokončí tyto aktivity, přidá skupiny vrstev do mapové kompozice. Ovládací prvek bude reagovat na změnu vyvolanou přidáním nových elementů do mapové kompozice a překreslí svůj obsah. Po dokončení procesu mapa obsahuje všechny mapové vrstvy, které jsou korektně rozřazeny do skupin vrstev, ovládací prvek „LayerSwitcher“ zobrazuje ty mapové vrstvy, které mají vlastnost showInLayerSwitcher a mapová kompozice je tedy korektně zobrazena uživateli.

Dalším procesem v pořadí je přiřazování události kliknutí na ty mapové vrstvy, které to umožňují. Od tohoto okamžiku se pracovalo nad kolekcí vrstev, která byla přidaná v objektu mapy a tudíž následující proces se již odehrává nad daty, které jsou přidány v mapové kompozici. Jelikož XML soubor obsahoval informaci o tom, která vrstva umožňuje dotazování, propagovala se tato vlastnost přes pomocné DTO entity až do samotných vrstev, kde se uložila jako přidaný atribut, kterým byly vlastnosti objektu rozšířeny. Pomocí tohoto atributu bylo posléze možné projít všechny vrstvy a vytvořit události kliknutí, které vykonávají funkci *GetFeatureInfo* a vracejí tak informace z GeoServeru o elementu, na kterém bylo provedeno kliknutí.

Dalšími kroky v implementaci bylo navázání dalších funkcí, jako například posun času, které manipulují s mapovou kompozicí. Pro tyto účely je v systému Floreon+ 3.0 nutné vždy volat funkce mapového objektu, aby vrátil požadované mapové vrstvy (pokud je má k dispozici). V předchozí verzi systému se explicitně dohledávala vrstva v mapové kompozici podle jména a tudíž mohlo docházet k vyhledávání vrstev, které v mapové kompozici vůbec nemusely být přidány. Ve výsledku je systém velmi závislý na korektním nastavení „back-endu“ aplikace a hlavně obsahu a nastavení dat. Výsledný systém omezil nutnost zásahu programátora při úpravě mapové kompozice na minimum, avšak vytvořil větší nároky na správce mapových vrstev na úrovni GeoServeru a případně WCF.

5.4.2 Mapování capabilities na DTO

Proces skládání mapové kompozice začíná voláním funkce *initDynamicLayers*, která je součástí hlavní funkce mapové kompozice *mapInit*. V této funkci je realizované Ajax volání WCF služby *GetAvailableLayers* a služby *GetSwitcherLayers*. Jak již bylo popsáno v dřívějších kapitolách, služba *GetAvailableLayers* potřebuje jako vstupní parametr uživatelský *token*, pomocí kterého přečte *capabilities* a vrátí seznam jmen všech vrstev dostupných pro uživatele. Služba *GetSwitcherLayers* žádné vstupní parametry nepotřebuje, jedná se totiž pouze o množinu, která slouží k vytvoření průniku s dostupnými vrstvami.

Jakmile jsou obě množiny načtené, ukládají se do globálních proměnných, tak aby se k nim dalo přistupovat v pozdější fázi procesu. Pokud proběhl proces volání *GetAvailableLayers* úspěšně, volá se funkce *callbackGetLayersForUsers*, která implementuje mapování kolekce DTO entit na kolekci vrstev. Jelikož ale v této fázi procesu ještě žádná kolekce DTO entit neexistuje, volá se funkce *createLayerStructure*.

Tato metoda se dá rozdělit na dvě části - část vytvoření kolekce DTO objektů a část, která načítá XML *capabilities* a postupně na každý objekt mapuje příslušné vlastnosti. Jako DTO objekt poslouží výchozí objekt *Object*, který v jazyce JavaScript lze libovolně rozšiřovat (tzn. takový objekt nemá pevně danou strukturu, ale každé nové přiřazení vlastnosti vytváří novou vlastnost objektu). Cyklus vytváření DTO objektů provede tolik iterací, kolik prvků má uložena kolekce *availableLayers*, kde pro každou iteraci vzniká nový objekt, který se na konci inicializace přidává do kolekce DTO objektů. Taktéž v každé iteraci zaznamená do pole jmenný prostor dané vrstvy. Toto pole se posléze využije k vytvoření skupin vrstev. Každý DTO objekt obsahuje název vrstvy, skupinu do které vrstva spadá, vlastnost „queryable“ a tedy, zda vrstva poskytuje informace po události kliknutí, název vrstvy v GeoServeru, zdroj dat (další objekt pojmenovaný *Source*) a informaci o zobrazení vrstvy. Informace o zobrazení vrstvy je ve výchozí implementaci nastaveno na „false“ a tedy, aby při načtení mapové kompozice nebyla vrstva zapnutá a vykreslená v mapové kompozici.

Zdroj dat je objekt, vytvořený naprosto stejným způsobem jako objekt celé DTO entity. Zdroj dat však je součástí DTO entity a reprezentuje atributy, které se posléze namapují na zdroj dat mapové vrstvy. Zdroj dat obsahuje atributy jako URL GeoServeru, název vrstvy v GeoServeru, atribut označující, zda se jedná o „tile“ WMS vrstvu, typ serveru (v našem případě GeoServer) a slovník „params“. Ve slovníku parametrů zdroje dat se specifikují další atributy, jako je požadovaný formát („image/png“) nebo styl vrstvy (ve výchozí implementaci prázdné). Taktéž se zde zapisuje informace o časové stopě, která se ve výchozím nastavení plní aktuálním časem. Důležitým parametrem je uživatelský autorizační klíč *token*, který taktéž musí být součásti požadavku o mapovou vrstvu na GeoServer. A další atributy, zda je vrstva základová, zda má být „tilovaná“ a pokud ne, tak jaké rozměry má mít celkový obrázek mapové vrstvy, apod. Struktura objektu je vytvořena tímto způsobem, neboť z velké části kopíruje strukturu objektů vrstev v OpenLayers. Po běhu cyklu na vytváření DTO entit, přejde metoda do druhé části, a to do procesu zpracování XML „capabilities“. Pomocí Ajax dotazu na službu *GetCapabilities*, kde celá URL je poskládaná z URL GeoServeru (v implementaci se jedná o proměnnou, ať lze snadno webového klienta připojit ke kterémukoliv z GeoServerů), parametru *service* s hodnotou WMS, verzí 1.3.0, *requestu GetCapabilities* a autorizačního *tokenu*, který je přiřazen uživateli. Odpověď je navrácena ve formátu XML a obsahuje celé *capabilities* podle zadaných parametrů.

Jelikož XML soubor je strukturálně stejný jako HTML, dá se procházet metodou XPath, avšak jQuery framework poskytuje funkce na efektivní procházení HTML dokumentu, nebo XML dokumentu. Víme tedy, že informace o vrstvách se nachází v rodičovském kořeni „Layer“,

který má potomky také s názvem „Layer“. V implementaci byl sestaven pomocí jQuery jednoduchý průchod těmito elementy - *$(xml).find('Layer').find('Layer')*, kde *xml* je proměnná s celým XML dokumentem *capabilities*. Můžeme vidět, že v prvním volání metody „find“ se spoléhá na skutečnost, že bude nalezen jeden element s tímto názvem, což bylo možné si v implementaci dovolit, neboť strukturu *capabilities* jsme dopředu již znali. V dalším volání metody „find“ se již nevyhledávalo v celém XML dokumentu, ale pouze v potomcích předešlého nalezeného elementu. Zde se však předpokládalo, že vrácená hodnota bude kolekce všech elementů, neboť nebylo blíže nespecifikováno, jaký element byl hledán a názvu „Layer“ v tomto případě odpovídaly všechny vrstvy. Vzhledem k faktu, že se jednalo o kolekci vrstev, mohla se na výsledek aplikovat funkce „each“, která vytvořila cyklus s takovým počtem iterací, kolik existovalo prvků v nalezené kolekci. Výsledná funkce procházející všechny elementy „Layer“ pak tedy vypadala následovně - *$(xml).find('Layer').find('Layer').each(function(){}*. Nacházeli jsme se tedy v cyklu, který umí projít všechny elementy „Layer“ z XML souboru, avšak bylo potřeba, aby se v rámci tohoto cyklu našel správný DTO element (který v rámci vytváření DTO kolekce již má přiřazené jméno vrstvy), na který se namapují informace. Znamená to tedy, že uvnitř tohoto cyklu musel být vytvořen druhý cyklus, který iteruje skrze kolekci DTO entit a porovnává atribut „name“ DTO entity s elementem „Name“, který je potomkem každé vrstvy v XML dokumentu. Dojde-li ke shodě, čte první cyklus element „Layer“ stejný, jako čte druhý cyklus DTO entitu. V takovém případě se namapují všechny relevantní informace z XML dokumentu na DTO entitu. Celý proces můžeme vidět zachycený ve výpisu kódu 2. Výsledkem procesu mapování XML *capabilities* na DTO entity byla tedy kolekce DTO entit, kde každá DTO entita reprezentovala jednu vrstvu mapové kompozice a nesla relevantní informace vrstvy, které jsou potřebné ke korektní vizualizaci v mapové kompozici na webovém rozhraní pomocí frameworku OpenLayers. Ve výsledku metoda vrací kolekci těchto již „naplněných“ DTO entit.

```
\$(xml).find('Layer').find('Layer').each(function () {

                for (var i = 0; i < layerObjectList.length; i++) {
                    if ($(this).children("Name").text() == layerObjectList[i].
                        geoNameLayer) {

                        var query = $(this).attr("queryable").toString();
                        layerObjectList[i].queryable = query;

                        var name = $(this).children("Name").text().toString();
                        var title = $(this).children("Title").text().toString();
                        var abstract = $(this).children("Abstract").text().
                            toString();
```

```
                var style = $(this).children("Style").children("Name").
                    text().toString();
                var crs = $(this).children("CRS").eq(0).text().toString()
                    ;
                var attributionCitation = $(this).children("Attribution")
                    .children("Title").text();
                if (attributionCitation != null &&
                    attributionCitation != undefined &&
                    attributionCitation !== "") {
                    layerObjectList[i].source.params.citation =
                        attributionCitation;
                } else {
                    attributionCitation = "";
                }
                layerObjectList[i].source.params.styles = style;
                layerObjectList[i].source.params.CRS = crs;
                layerObjectList[i].source.params.CITATION =
                    attributionCitation;

            }
        }
    });
```

Výpis 2: Proces vytváření DTO objektů vrstev

5.4.3 Mapování DTO na vrstvy

Dostali jsme se do části implementace procesu, kde na vstupu byla kolekce DTO entit a chtěli jsme z nich vytvořit objekty třídy *ol.layer.Tile*, čímž vzniknou mapové vrstvy. V tomto procesu se musely přetransformovat DTO entity na objekty vrstev (příklad mapování jedné vrstvy lze vidět v ukázce kódu 3) a navíc vytvořit průnik množin všech vrstev a vrstev, které se mají zobrazovat v „LayerSwitcheru“. Celá metoda se opět odehrává v cyklu, který iteruje kolekci DTO entit a vytváří postupně objekty vrstev. Prakticky se takto vytváří objekty mapových vrstev a objekty zdrojů dat (*ol.source.TileWMS*). Na zdroj dat se posléze přemapují všechny atributy z objektu zdroje dat jednotlivých DTO entit a na objekt mapové vrstvy ty atributy, které jsou přímo v entitě DTO.

```
var s = new ol.source.TileWMS({
            url: data[i].source.url,
            params: {
                "STYLES": data[i].source.params.styles,
                "LAYERS": data[i].source.params.layers,
                "FORMAT": data[i].source.params.format,
                "TIME": data[i].source.params.time,
                "AUTHKEY": data[i].source.params.authkey,
                "WIDTH": data[i].source.params.width,
                "HEIGHT": data[i].source.params.height,
                "ISBASELAYER": data[i].source.params.isBaseLayer,
                "TRANSPARENT": data[i].source.params.transparent,
                "TILED": data[i].source.params.tiled,
                "CRS": data[i].source.params.crs,
                "CITATION": data[i].source.params.citation,
            },
            serverType: data[i].source.serverType,
            crossOrigin: data[i].source.crossOrigin,
            authkey: data[i].source.authkey
        });

        var l = new ol.layer.Tile({
            queryable: data[i].queryable,
            title: data[i].name,
            source: s,
            visible: data[i].visible,
            geoNameLayer: data[i].geoNameLayer,
            group: data[i].group

        });
```

Výpis 3: Mapování DTO entity na objekt vrstvy

Zároveň se explicitně nastavil přidaný atribut *showInLayersSwitcher* na hodnotu „false“, neboť proces filtrování vrstev, které se mají zobrazovat v ovladači vrstev byl implementován až v další fázi. Po dokončení cyklu, ve kterém se mapovaly objekty vrstev, přišel na řadu proces vytvoření skupin vrstev. Již v procesu vytváření DTO entit bylo vytvořilo pole, které se plnilo vždy jmenným prostorem zpracovávané vrstvy. Tento seznam měl tedy v této fázi procesu stejnou velikost jako kolekce vrstev, avšak obsahoval mnoho duplicit. Bylo potřeba pro každý

unikátní jmenný prostor vytvořit objekt *ol.layer.Group*, do kterého se posléze shlukovaly vrstvy se stejným jmenným prostorem. Byla tedy implementována funkce, která odstraňuje duplicity a vrací seznam unikátních hodnot v poli. Tato funkce se posléze aplikovala na celý seznam, který obsahoval jmenný prostor každé vrstvy a výsledkem byla množina unikátních jmenných prostorů, resp. skupin mapových vrstev. V cyklu byla vytvořena pro každý unikátní záznam nová skupina vrstev, která byla pojmenována podle daného jmenného prostoru. V této fázi implementace nebyla skupina vrstev přidávána do mapové kompozice, ale byla pouze uložená v proměnné.

Byli jsme ve fázi procesu, kdy bylo potřeba roztřídit vrstvy, které se mají zobrazovat v „LayerSwitcheru“ a vrstvy, které mají být přidány do mapové kompozice jako „pomocné“. Implementace se skládala ze tří cyklů, navzájem do sebe vložených, kde první cyklus iteruje skrze všechny vrstvy v seznamu vrstev, které mají být zobrazeny v „LayerSwitcheru“. Druhý cyklus iteruje skrze všechny vrstvy, které jsou načtené v kolekci vrstev a třetí cyklus iteruje skrze kolekci vytvořených skupin vrstev. Ukázku tohoto kódu můžeme vidět ve výpisu 4.

```
for (var i = 0; i < layerForSwitcher.length; i++) {
          for (var j = 0; j < dynamicLayers.length; j++) {
              if (layerForSwitcher[i] == dynamicLayers[j].get('geoNameLayer'))
                   {
                  for (var k = 0; k < allGroupsNames.length; k++) {
                      if (allGroupsNames[k] == dynamicLayers[j].get('group')) {
                          dynamicLayers[j].setProperties({ "showInLayerSwitcher
                              ": true });
                          addedByGroup.push(dynamicLayers[j]);
                          allGroups[k].getLayers().push(dynamicLayers[j]);

                      }
                  }
              }
          }
      }
```

Výpis 4: Třídění vrstev do mapové kompozice

V prvním cyklu se tedy vybere jedna z vrstev, které mají být zobrazeny v ovládacím prvku a v druhém se vyhledává tatáž vrstva. Jakmile dojde ke shodě, je zřejmé, že tato vrstva má být zobrazena v ovládacím prvku kompozice a tudíž i musí mít vlastní skupinu vrstev. V třetím cyklu probíhá iterace skrze skupiny vrstev a hledá shodu s jmenným prostorem zkoumané vrstvy. V momentě shody je daná vrstva přiřazena do vybrané skupiny a atribut vrstvy *showInLayerSwitcher* je nastaven na hodnotu „true“. Taktéž se daná vrstva odebere z kolekce všech načtených vrstev. Cyklus se opakuje tak dlouho, dokud nejsou všechny vrstvy roztříděné a kolekce zná-

mých vrstev obsahuje jen ty vrstvy, které nemají být zobrazeny v ovládacím prvku. Zbývalo přidat do skupin vrstev ty vrstvy, které jsou v mapové kompozici jako „pomocné“. Byl proto implementován cyklus skrze zbývající vrstvy, který porovnává názvy skupin vrstev a v případě shody přidává danou vrstvu do vybrané skupiny vrstev, avšak s atributem *showInLayerSwitcher* nastaveným na „false“.

Ve finální fázi bylo potřeba skupiny vrstev přidat do mapové kompozice a zkontrolovat, zda mapová kompozice korektně obsahuje všechny vrstvy, skupiny vrstev a také, zda vytvořený ovládací prvek správně zobrazuje mapové vrstvy a skupiny vrstev. Výsledkům implementace se dále věnuje kapitola 6.

5.4.4 Události nad vrstvami

Tato část textu se věnuje funkci systému, která je založena na události v mapové kompozici. Konkrétněji si můžeme představit kliknutí na element v mapové kompozici, neboť se jedná o nejčastější událost v mapové kompozici. Pochopitelně, událost kliknutí ve vrstvě má smysl pouze pro ty vrstvy, které poskytují na základě kliknutí na element data. Jistě si vzpomeneme, že v XML *capabilities* byly tyto vrstvy označené atributem „queryable“ a že tuto vlastnost vrstev jsme v řešení propagovali až do objektů vrstev samotných. Znamená to tedy, že ve mapové kompozici byla k dispozici informace, na základě které, bylo možné vytvořit událost kliknutí pro ty mapové vrstvy, u kterých má tato událost smysl. Pro názornost si uveďme několik příkladů - vezměme si mapovou vrstvu říční sítě a vrstvu dopravních informací. Vrstva říční sítě po zviditelnění v mapové kompozici vizualizuje vodní plochy a toky, kde vrstva neobsahuje po kliknutí na vodní tok, žádné dodatečné informace. Vrstva dopravních informací obsahuje ikonky znázorňující omezení na dopravních úsecích. Zatímco vrstva říční sítě a vodních ploch nese informaci sama o sobě, vrstva dopravních informací by nám ve stejném případě pouze dala informaci o umístění „nějakého omezení“ dopravy. U vrstvy dopravních informací zcela přirozeně očekáváme další informace k jednotlivým místům omezení dopravy, které získáme kliknutím na ikonku omezení v mapové kompozici. V systému Floreon+ 2.0 byl tento úkol triviální, avšak byl implementován statickým způsobem, což mohlo způsobovat komplikace. To znamená, že v případě přidání nové mapové vrstvy se explicitně ve zdrojovém kódu definovalo chování při události kliknutí, které obsahovalo umístění, kam měl dotaz směřovat a jak se mají získaná data ve webovém rozhraní zobrazit. Zůstaneme-li u dopravních informací, tak v předchozí verzi systému byla definovaná událost kliknutí na mapovou vrstvu dopravních informací, kdy při kliknutí byla vyvolaná funkce, které se předávala informace o který element v mapě se jednalo. Tato funkce zavolala službu *GetFeatureInfo* na GeoServeru s parametry dané vrstvy a dostala zpět odpověď v požadovaném formátu (konkrétně ve formátu JSON). V hlavním HTML souboru byla definovaná struktura, jak data o dopravní informaci zobrazit a tudíž funkce pouze nahradila vybrané oblasti v této struktuře daty a celou strukturu zobrazila v pravém panelu. Ačkoliv se jedná o triviální implementaci, můžeme takovýto úkol označit za komplikovaný z důvodu mnoha nutných zásahů programátora na mnoha místech v kódu. Programátor v případě přidání mapové

vrstvy musel definovat strukturu v HTML dokumentu, zpracovat JSON odpověď a korektně ji zformátovat na vytvořenou strukturu a k tomu všemu kontrolovat práva, zda uživatel vůbec na tuto akci má oprávnění.

V rámci předělání systému na dynamický přístup (skládání mapové kompozice) bylo nutné implementovat i tuto část dynamicky. Vznikla tedy nutnost vymyslet a implementovat funkčnost kliknutí do mapy takovým způsobem, aby se manuálně nemusely struktury odpovědi tvořit ve webovém rozhraní, ale aby byly součástí odpovědi GeoServeru. Návrh řešení je poměrně snadný. Vzhledem ke skutečnosti, že GeoServer v požadavku *GetFeatureInfo* umožňuje požadovat odpověď ve formátu HTML, je možné implementovat dotaz dynamicky, kdy ale neočekáváme pouze surová data, které teprve musíme korektně umístit do „šablony“ vytvořené v HTML, ale očekáváme rovnou strukturu HTML s daty. GeoServer umí pro své odpovědi definovat HTML strukturu, avšak je nutností tyto šablony udržovat aktuální vzhledem k informacím, které se mají předat a taky je zde skutečnost, že se jedná o další zodpovědnost přenesenou z webového rozhraní na „back-end“ aplikace.

Vraťme se tedy k vytvoření události kliknutí na vrstvy, jež tuto akci podporují. Počátek události je tedy zcela logicky v momentě kliknutí na objekt mapové kompozice. Tuto událost šlo vytvořit poměrně jednoduše pomoci jQuery. Vytvoření události na mapové kompozici bylo tedy implementováno pomocí *map.on('click', function(e)){}*, kde *e* označuje samotnou událost. Dále byla implementována funkce *checkFeatureClickEvent*, která jako vstupní parametr bere právě objekt události (tedy v případě této implementace objekt označený jako *e*).

V implementované funkci bylo již možné přistupovat k mapové kompozici dynamickým způsobem. Chtěli-li jsme tedy vybrat všechny vrstvy, jejichž parametr „queryable“ má hodnotu „true“, museli jsme si z mapové kompozice nejdříve vytáhnout všechny vrstvy. To lze provést funkcí *getLayers*, kterou implementuje objekt *map*. Pomocí této funkce lze získat všechny vrstvy, které mapová kompozice má k dispozici. Následně se v procesu iterovalo skrze všechny tyto vrstvy a kontrolovalo se, zda každá vrstva je viditelná v kompozici (funkce *getVisible* vrstvy), zda vrstva má vytvořený atribut „queryable“ nastavený na „true“ a také, zda se nejedná o základovou vrstvu, neboť žádná ze základových vrstev nevrací data po kliknutí (a ani v korektním nastavení nemá). Jakmile se v rámci iterace narazí na vrstvu, která splňuje všechny podmínky, vytáhne se z atributů vrstvy URL směřující na *FeatureInfo* dané vrstvy. Jedná se o jeden z atributů přenesených z XML *capabilities* při mapování. Na vytvořené URL se vykoná Ajax volání, které očekává v odpovědi HTML strukturu s daty. V případě, že data jsou k dispozici a korektně navráceny webovému rozhraní, probíhá proces zpracování HTML odpovědi na DOM elementy a přidání do kolekce *contentBuffer*. Tato část nebude v tomto textu příliš rozvíjena, není relevantní k tématu dynamické mapové kompozice. Ve zkratce se tedy obsah dat, již v připravené HTML struktuře, uloží do kolekce, která se posléze stane obsahem kontextového menu v mapové kompozici. Uživatel si z kontextového menu vybere položku, kterou chce vizualizovat a ta se posléze zobrazí v pravém panelu s informacemi a již v HTML struktuře. Kontextové menu bylo implementováno z důvodu možného překryvu elementu v mapové kompozici a vytvoření

možnosti pro uživatele vybrat si, o které informace má zájem.

Některé vrstvy mají v GeoServeru definovaný styl nazvaný „selected“. Je-li na tyto vrstvy vykonána událost kliknutí, lze pomocí CQL filtru (způsob, jak vybrat z mapové vrstvy pouze některé elementy) vybrat ty mapové elementy, které byly kliknutím „zasaženy“ a na tyto elementy aplikovat daný styl. Výsledný efekt vytváří dojem, že kliknutím do mapy se vybral dotazovaný element z kompozice. Ve chvíli kdy uživatel zavře informace získané z elementu v pravém panelu, nebo klikne „bokem“ v mapové kompozici, aplikovaný styl se z vrstvy odebere a mapová kompozice se nachází ve výchozím stavu.

5.4.5 Interakce s rozhraním

Interakce mapové kompozice a webového rozhraní byla narušena více z důvodu přechodu na novou verzi OpenLayers, než z důvodu implementace dynamické kompozice. Tato myšlenka je rozvedena dále, prvně ale uveďme pár příkladů, co je považováno za interakci s mapovou kompozicí, kromě prvků, o kterých jsme již mluvili. Příklady interakce mapové kompozice a rozhraní může být například posun času, nástroje měření v mapové kompozici nebo dokonce vytváření analýz a predikcí.

```
function updateLayersTime() {
    getAllLayersFromMap()
        .forEach(function (l) {
            l.getSource().updateParams({ 'TIME': getTimeStamp() });
        });
}
```

Výpis 5: Změna času vrstev

Vzhledem k tomu, že v předchozí verzi systému se mapová vrstva vždy vybírala pomocí jména vrstvy (*map.getLayerByName(název vrstvy)*), nelze stejným způsobem přistupovat k dynamicky sestavené kompozici. Respektive nová verze OpenLayers funkci vyhledání mapové vrstvy podle názvu implementuje, avšak nelze se spoléhat, že vrstva s požadovaným názvem se v dané kompozici nalézá. Znamená to tedy, že v každém případě, kdy je potřeba vyhledat objekt vrstvy v mapové kompozici a s tímto objektem dále pracovat, je nutné projít všechny vrstvy v mapové kompozici, zda je vybraná vrstva přítomná a dostupná. Z tohoto důvodu byly implementovány funkce pro vybírání vrstev z mapové kompozice, pro přidávání a odebírání nových vrstev (vzniklých za běhu webového klienta) a spousta dalších podpůrných funkcí. Je-li potřeba tedy do interakce s rozhraním zapojit konkrétní vrstvu, zavolá se tato funkce (implementována jako *getLayerByName*) a pokud je vrstva nalezena, je funkcí vrácena, v opačném případě vrací metoda prázdnou hodnotu. Všechny funkcionality ze systému Floreon+ 2.0 se tedy implementují analogicky, pouze s tímto rozdílným přístupem k vrstvám. Budeme-li tedy mluvit například o posunu času ve webovém rozhraní, musela se implementovat událost posunu času, kdy se při

této události spouští metoda, která projde všechny vrstvy v mapové kompozici a změní v atributech každé vrstvy časovou stopu. Implementaci této jednoduché funkce lze vidět v ukázce kódu ve výpisu 5. OpenLayers implementuje na takovou změnu událost, která již sama zajišťuje obnovení dat ze zdroje s již novou časovou stopou. Výsledkem je tedy opravdu časový posun v prostorových datech, pokud jsou data dostupná.

Taktéž je nutné zmínit, že v nové verzi Floreon+ 3.0 se všechny mapové vrstvy z GeoServeru implementují jako WMS a jako WFS se používají pouze pomocné vrstvy, které většinou vznikají za běhu webového klienta aplikace. Důvodem je samotný princip dat. Uveďme si příklad na vrstvě měřících stanic. V případě WFS se v mapové kompozici zobrazují pouze body (resp. geometrie) reprezentující umístění měřících stanic a v systému Floreon+ 2.0 posléze na tyto body algoritmus mapuje SVG obrázky s názvem stanice a povodňovým stupněm. Plánem je sjednotit veškeré vrstvy, které to umožní, pod standard WMS a případné operace provádět posléze pomocí WPS služby (jedná se o rozšíření GeoServeru o služby poskytující prostorové procesy, algoritmy a kalkulace). V této souvislosti v některých případech odpadá například výpočet průniků dvou různých vrstev při zadávání analýzy, ale pouze o vytvoření požadované oblasti a odeslaní této geometrie spolu se zadáním analýzy nebo predikce.

5.4.6 Lokalizace

Poslední částí dynamické koncepce projektu byla „lokalizace“. „Lokalizace“, neboli mutace webového rozhraní do více jazyků byla v systému Floreon+ 2.0 implementována pomocí metody, která se spouštěla po načtení webového rozhraní. Jakýkoliv textový element v celém webovém rozhraní je v HTML kódu napsán „tagem“, rozumějme však tomuto označení jako textu, který je klíčem pro pozdější náhradu podle výběru lokalizace. Znamená to tedy, že dokument HTML je plný pouze klíčových slov a je tedy plně nezávislý na jazyku. Každý text, který má být posléze nahrazen správným výrazem ze zvoleného jazyka, musí náležet HTML elementu, který má přiřazenou třídu *lang*. Toto označení je nejdůležitější pro algoritmus mutace rozhraní.

Algoritmus mutace rozhraní do vybraného jazyka se spouští kompletním načtením webového rozhraní a hledá v HTML dokumentu všechny elementy označeny třídou *lang*. Jakmile takový element najde, přečte algoritmus vnitřní textový klíč a podle tohoto klíče vyhledává ve svém definovaném slovníku. Jakmile nalezne hledaný klíč ve slovníku, vybere přiřazený výraz a klíčové slovo v HTML dokumentu nahradí tímto výrazem. Slovník je dvojrozměrné pole implementované v JavaScriptu a znamená to tedy, že slovník je udržován na úrovni webového klienta.

Bylo potřeba uvědomit si, že kvůli dynamické mapové kompozici se musím vytvořit nový koncept implementace (dynamické vytváření DOM elementů namísto plnění předem vytvořených šablon) a že již není možné nadále udržovat slovník na úrovni webového klienta. Respektive je to možné, ale znamená to omezení dynamického konceptu vytváření obsahu na webovém rozhraní. Důvod lze demonstrovat na následující ukázce. Představme si, že na úrovni GeoServeru se publikuje nová vrstva, která má atribut „queryable“ (a lze tedy po kliku na element ve vrstvě dotazovat další data). Na úrovni GeoServeru se tedy musí specifikovat i struktura výsledku

dotazu po kliknutí a pochopitelně data, která se mají vracet. Pokud jsou tato data jazykově závislá, je vhodné tato data také nahradit klíčem a vytvořit k danému klíči výrazy ve všech jazycích, které webové rozhraní podporuje. Pokud by slovník zůstal na úrovni webového klienta, znamenalo by to, že se musí zasahovat do kódu webového rozhraní a slovník modifikovat. Ve chvíli, kdy by slovník obsahoval požadovaný výraz, mohl by proces fungovat takovým způsobem, že při přijetí odpovědi z GeoServeru se odpověď vloží do algoritmu na překládání a až přeložený výsledek se vizualizuje.

Zde lze tedy pozorovat potřebu přesunutí slovníku také na „back-end“ aplikace. V takovém případě pak správce, který vytvoří novou mapovou vrstvu a strukturu dat, které se k dané vrstvě vracejí při dotazu, musí vytvořit ještě záznamy použitých výrazů do slovníku. Pouze takovým způsobem lze modifikování mapové kompozice zcela eliminovat od nutnosti úprav kódu webového rozhraní. V systému Floreon+ 3.0 se tedy implementuje tento slovník na úrovni MS-SQL databáze, kde při načítání webového rozhraní se slovník načte spolu s rozhraním pomocí volání Ajax. WCF služba implementuje jednoduchou metodu, která vrátí celou tabulku se slovníkem ve formátu JSON. Po přijetí slovníku v odpovědi se spouští algoritmus, který slovník přetvoří do stejného dvojrozměrného pole, jako tomu bylo u systému Floreon+ 2.0 a dále následný proces je již stejný.

6 Testování řešení

Tato část textu se věnuje testování vyvinuté části systému Floreon+ 3.0. Hlavním předmětem testů bylo ověření správnosti generování mapové kompozice pro uživatele s odlišnými rolemi v systému. Dalším předmětem testování bylo fungování ovládacího prvku mapové kompozice a události v mapové kompozici. Všechny takto vykonávané testy jsou funkcionální a byly tedy prováděny principem spouštění aplikace a prováděním předem definovaných úkonů.

V rámci systémového testování byly provedeny tři různé testovací scénáře. První dva scénáře mají předpokládaný výsledek, skutečný výsledek a případně údaj, zda se množiny vrstev shodují. Poslední scénář obsahuje testovací kroky, případné vstupní data nebo podmínky testu, očekávaný výsledek a výsledek testovacího kroku. Na konci každého scénáře je slovní vyhodnocení daného testu.

Pro potřeby testů vzniklo celkem sedm nových uživatelů, kterým byly přiřazeny rozdílné role v systému. Jedná se o tyto uživatele (e-mail slouží jako přihlašovací údaj):

- Test.admin@it4i.cz - role „administrátor“
- Test.basic@it4i.cz - role „basic“
- Test.expert@it4i.cz - role „expert“
- Test.rodos@it4i.cz - role „rodos“
- Test.pcr@it4i.cz - role „prezidium pcr“
- Test.gx@it4i.cz - role „gx“
- Test.hzsmsk@it4i.cz - role „hzsmsk“

6.1 Testování mapové kompozice

Cílem tohoto testu bylo zjistit, zda se mapová kompozice skládá pro různé role uživatelů z různých vrstev (rozdílného počtu vrstev). Jako předpokládaný výsledek posloužila množina vrstev z XML souboru *capabilities* pro každého uživatele s průnikem vrstev, které omezuje WCF služba. Pro účely testů bylo vytvořeno znovu volání služby *GetAvailableLayers*, která vrací omezenou množinu vrstev z XML *capabilities*. Úplně stejně bylo pro druhý test implementováno volání služby *GetSwitcherLayers*, která vrací všechny vrstvy, které mají být zobrazeny v ovládacím prvku „LayerSwitcher“. Předpokládaná množina vrstev přidaných do mapové kompozice se tedy skládá z množiny vrstev ze souboru *capabilities* a množiny vrstev vrácených službou *GetAvailableLayers*.

Pro účely tohoto testu byl implementován algoritmus, který po načtení mapové kompozice uživatele vypsal všechny vrstvy obsažené v mapové kompozici do konzole prohlížeče. Podmínkou splnění testu byla shoda všech názvů vrstev v obou množinách pro různé uživatelské role.

Uživatel	Vrstvy (předpoklad)	Vrstvy (skutečnost)	Rovnost množin
Test.admin	44	44	rovnost
Test.basic	14	14	rovnost
Test.expert	14	14	rovnost
Test.rodos	16	16	rovnost
Test.pcr	19	19	rovnost
Test.gx	19	19	rovnost
Test.hzsmsk	27	27	rovnost
Nepřihlášený	23	23	rovnost

Tabulka 1: Test obsahu mapové kompozice

Z testovací tabulky můžeme vyčíst, že testování všech vybraných uživatelských rolí dopadlo v pořádku. Obsah mapové kompozice pro každou roli odpovídal mapovým vrstvám vystavených pro danou roli. Můžeme si tedy všimnout, že počet očekávaných vrstev v mapové kompozici odpovídal skutečnému počtu vrstev přidaných do mapové kompozice a zároveň, že tyto dvě množiny obsahovaly stejné prvky.

6.2 Testování ovládacího prvku

Cílem tohoto testu bylo zjistit, zda implementovaný ovládací prvek mapové kompozice plní správně svou roli. Předpoklad pro tento test byl úspěšný předchozí test mapové kompozice. Z testu mapové kompozice posloužily výsledky jako množina vrstev, která měla být v mapové kompozici přidána. Druhou množinou vrstev byla množina vrácená funkcí *GetSwitcherLayers*. Průnik těchto dvou množin vytvořil předpokládanou množinu vrstev, která měla být zobrazena v „LayerSwitcheru“. Skutečný výsledek (množina zobrazených vrstev) byl posléze vypozorován z chování „LayerSwitcheru“ pro různé skladby mapové kompozice.

Uživatel	Vrstvy (předpoklad)	Vrstvy (skutečnost)	Rovnost množin
Test.admin	22	22	rovnost (*)
Test.basic	14	14	rovnost
Test.expert	14	14	rovnost
Test.rodos	14	14	rovnost
Test.pcr	16	16	rovnost
Test.gx	17	17	rovnost
Test.hzsmsk	22	22	rovnost
Nepřihlášený	19	19	rovnost

Tabulka 2: Test ovládacího prvku kompozice

Test ovládacího prvku dopadl také v pořádku, až na jednu odchylku od očekávaného chování. U uživatele „test.admin“ byla tato odchylka zaznačena hvězdičkou, neboť počet vrstev

v ovládacím prvku odpovídal počtu vrstev, které se zobrazit měly, avšak neodpovídal počet zobrazených skupin vrstev. Pro tohoto uživatele existovala skupina vrstev „Znečištění“, ale neexistovala žádná vrstva, která by byla v této skupině zobrazená. Důvodem toho chování je nekonzistentní nastavení uživatele „Switcher“, který je definovaný na GeoServeru a používá se k definici vrstev, které se mají zobrazovat v ovládacím prvku. Znamená to tedy, že například pro uživatele „test.admin“ pak nastávají situace, kdy uživatel má oprávnění vrstvu vizualizovat a ovládat, avšak jelikož vrstva není v definici vrstev, které se mají zobrazovat v „LayerSwitcheru“, tak se vrstva uživateli nezobrazí.

Z tohoto důvodu můžeme tuto chybu přisuzovat nesprávnému nastavení uživatelských rolí, které je pro dynamickou kompozici kritické. Taktéž je zde nutnost přiznat, že zde vznikl prostor pro vylepšení. Vylepšení by spočívalo například ve zneviditelnění skupiny vrstev, pokud by došlo k takovémuto nekonzistentnímu nastavení.

U ostatních uživatelů obsah ovládacího prvku odpovídal počtu vrstev, které podle nastavení měly být v ovládacím prvku zobrazeny.

6.3 Testování událostí v mapové kompozici

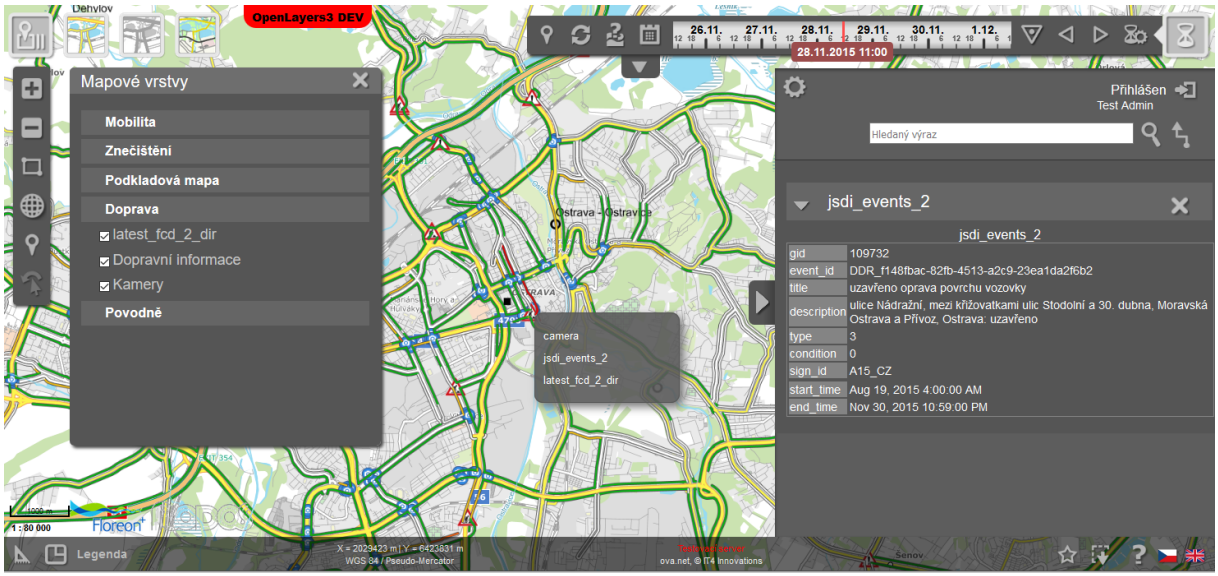
Cílem tohoto testu bylo ověřit správné fungování vytvořených událostí kliknutí do mapové kompozice. Testování mohlo být realizováno v libovolně zvolené mapové kompozici, avšak se znalostí vrstev, které implementují událost kliknutí a poskytují tak data. Reálný výsledek byl posléze pozorován v mapové kompozici, zda při kliknutí do kompozice a „zasažení“ kliknutím elementu vrstvy, která událost kliknutí implementuje, správně systém vizualizoval získaná data. V případě kolize více datových toků (kliknutí do mapy „zasáhlo“ elementy z více mapových vrstev) bylo pozorováno, zda systém správně zobrazil kontextové menu.

Pro účely tohoto testu byl zvolen uživatel „test.admin“ z důvodu největšího počtu vrstev v mapové kompozici. Vzhledem k tomu, že při implementaci bylo omezeno vytváření události kliknutí jen na mapové vrstvy, které nejsou podkladové (skupina vrstev „basemaps“) a množina vrstev „LayerSwitcher“ kompozici taky zúžila, zůstaly v mapové kompozici celkem čtyři relevantní vrstvy k této události. Jsou to vrstvy dopravních kamer, dopravních informací, vytížení dopravních uzlů a měřících stanic. Aby se docílilo překryvu elementů vrstev, byl posunut čas v rozhraní na 28.11.2015, neboť zde existují překryvy elementů ve vrstvách. Následující seznam kroků zachycuje průběh testu.

1. Zobrazení vrstev dopravních kamer, dopravních informací, vytížení dopravních uzlů a měřících stanic - OK
2. Kliknutí mimo elementy všech vrstev, neočekávaná žádná událost - OK
3. Kliknutí na element jedné vrstvy, očekávána událost - OK
4. Zavření zobrazené události - OK

5. Kliknutí na více překrytých elementů, očekáváno kontextové menu - OK
6. Výběr z kontextového menu, očekáváno zobrazení vybraných dat - OK
7. Kliknutí na element kamery v kompozici, očekáváno přidání informací k již zobrazeným informacím z předchozího kroku - OK
8. Kliknutí na jiné kamery v kompozici, než v předchozím kroku, očekáváno přepsání zobrazených informací k danému elementu - OK

Z výsledku testu můžeme vyčíst, že události v mapové kompozici fungují v pořádku. Ať už se kliknutí týká jedné mapové vrstvy nebo více mapových vrstev, rozhraní umí s události zacházet. V případě překryvu elementů nabízí mapová kompozice výběr z kontextového menu (lze vidět na obrázku 9). Struktura vzhledu zobrazených informací v pravém panelu se přebírá z šablony pro daná data z GeoServeru. Mimo jiné byl v tomto testu vyzkoušen i posun času v mapovém rozhraní, který taktéž v pořádku funguje a plní svou funkčnost.



Obrázek 9: Událost kliknutí do mapové kompozice

Na obrázku 9 taktéž můžeme vidět, že lokalizace prostředí není úplná z důvodu absence klíčových slov a jejich překladů pro některá hesla. Pro vyřešení tohoto problému stačí přidat příslušná hesla do slovníku pojmů. Celkově tedy test lze považovat za úspěšný a implementaci interakce rozhraní a mapové kompozice lze považovat za funkční.

7 Závěr

Webový klient systému Floreon+ prošel výraznými úpravami, ve kterých došlo v podstatě ke změně základního fungování webového rozhraní tohoto systému. Za nejdůležitější změny můžeme považovat aktualizaci mapového frameworku OpenLayers a vytvoření dynamického konceptu celého webového klienta. Aktualizace OpenLayers byl proces přidružený k této práci, neboť při takto rozsáhlém zásahu je vhodné přejít na nejnovější verzi používaného mapového frameworku. Nově vyvinutá dynamická mapová kompozice umožňuje snazší rozšiřitelnost systému o nové mapové vrstvy i skupiny vrstev. Pro účely dynamické mapové kompozice se musel změnit koncept celého webového rozhraní. Tento koncept staví na vytváření částí webového rozhraní dynamickým způsobem. Za zásadní může být považováno, aby se v dalším vývoji webového rozhraní pokračovalo v tomto konceptu. Je potřeba při dalším vývoji uvažovat, zda nově implementovaná funkcionalita bude vždy přístupná všem uživatelům systému a jestli je tedy vhodné implementovat funkcionalitu staticky.

Podíváme-li se však komplexně na novou implementaci, jistě nalezneme několik kritických bodů. Vzhledem ke skutečnosti, že se téměř veškerá odpovědnost za obsah mapové kompozice přenesla na úroveň GeoServeru, prostorové databáze a služeb implementovaných pomocí WCF, je nutné udržovat tato prostředí aktuální a správně nastavená. Pokud vznikne v systému nekonzistentní nastavení, pak nelze očekávat správné fungování mapové kompozice, ani funkcionalit spjatých s mapovou kompozicí. Částečně se přenesla i zodpovědnost za vzhled některých částí systému, konkrétně zobrazování informací relevantních k mapové vrstvě po kliknutí na element vrstvy. Struktura dat se také získává z GeoServeru, kde je nutné mít udržovanou šablonu pro data, která se zobrazují ve webovém rozhraní systému Floreon+ 3.0. Dalším kritickým bodem může být lokalizace rozhraní. Jakmile vznikne nová mapová vrstva, musí se vytvořit i všechny překlady v MS-SQL databázi, aby se všechny informace k dané vrstvě zobrazovaly ve webovém rozhraní korektně.

Naopak nespornou výhodou v procesu přidávání nové mapové vrstvy je absence programátora. Dalším rozšířením systému by mohla být například implementace dynamického sestavování rozhraní pro analýzy a predikce podle práv uživatele, což je zatím část systému, která je pevně vytvořena v kódu. Taktéž by se rozšíření systému mohlo posunout na celorepublikovou úroveň a implementovat tak funkcionality pro podporu krizového řízení v celé České republice. Věřím, že implementace systému Floreon+ 3.0 přispěla velkým měřítkem k snadnému rozšiřování systému a v brzké době se systém stane nenahraditelným nástrojem v oblasti krizového řízení.

Literatura

[1] OpenLayers 3 API Documentation [online]. [cit. 2017-04-28]. Dostupné z: http://openlayers.org/en/v3.0.0/apidoc/

[2] GeoServer Documentation [online]. [cit. 2017-04-28]. Dostupné z: http://docs.geoserver.org/

[3] An Esri(R) White Paper. Esri Support for Geospatial Standards: OGC and ISO/TC211 [online]. 2015 [cit. 2017-04-28]. Dostupné z: http://www.esri.com/library/whitepapers/pdfs/supported-ogc-iso-standards.pdf

[4] YOUNGBLOOD, Brian a Stefano IACOVELLA. GeoServer Beginner's Guide: Share and edit geospatial data with this open source software server. Birmingham: Packt Publishing Limited, 2013. ISBN 9781849516686.

[5] GRATIER, Thomas, Paul SPENCER a Erik HAZZARD. OpenLayers 3 Beginner's Guide: Get started with OpenLayers 3 and enhance your web pages by creating and displaying dynamic maps. Second published: January 2015. Birmingham: Packt Publishing Limited, 2015. ISBN 9781782162360.

[6] Fedorčák D., Kocyan T., Hájek M., Szturcová D., Martinovič J. (2014) viaRODOS: Monitoring and Visualisation of Current Traffic Situation on Highways. In: Saeed K., Snášel V. (eds) Computer Information Systems and Industrial Management. CISIM 2014. Lecture Notes in Computer Science, vol 8838. Springer, Berlin, Heidelberg

A Příloha na CD

Na přiloženém disku CD lze nalézt následující soubory.

1. Obrázky z průběhu testování řešení
2. Částí kódu z implementace řešení
3. Soubor popisující jak spustit implementované řešení
4. Digitální verze této práce ve formátu PDF